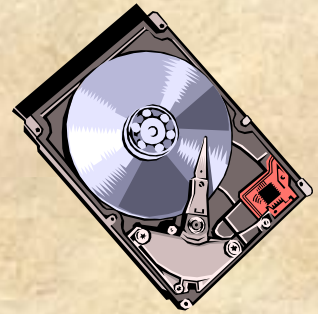
Operating Systems: Internals and Design Principles

Chapter 11 I/O Management and Disk Scheduling

Seventh Edition
By William Stallings

Categories of I/O Devices

External devices that engage in I/O with computer systems can be grouped into three categories:



Human readable

- suitable for communicating with the computer user
- printers, terminals, video display, keyboard, mouse

Machine readable

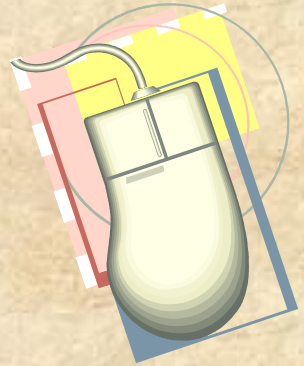
- suitable for communicating with electronic equipment
- disk drives, USB keys, sensors, controllers

Communication

- suitable for communicating with remote devices
- modems, digital line drivers

Differences in I/O Devices

- Devices differ in a number of areas:



Data Rate

- there may be differences of magnitude between the data transfer rates

Application

- the use to which a device is put has an influence on the software

Complexity of Control

- the effect on the operating system is filtered by the complexity of the I/O module that controls the device

Unit of Transfer

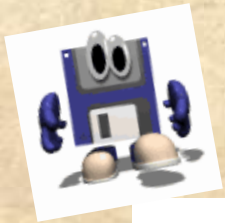
- data may be transferred as a stream of bytes or characters or in larger blocks

Data Representation

- different data encoding schemes are used by different devices

Error Conditions

- the nature of errors, the way in which they are reported, their consequences, and the available range of responses differs from one device to another



Data Rates

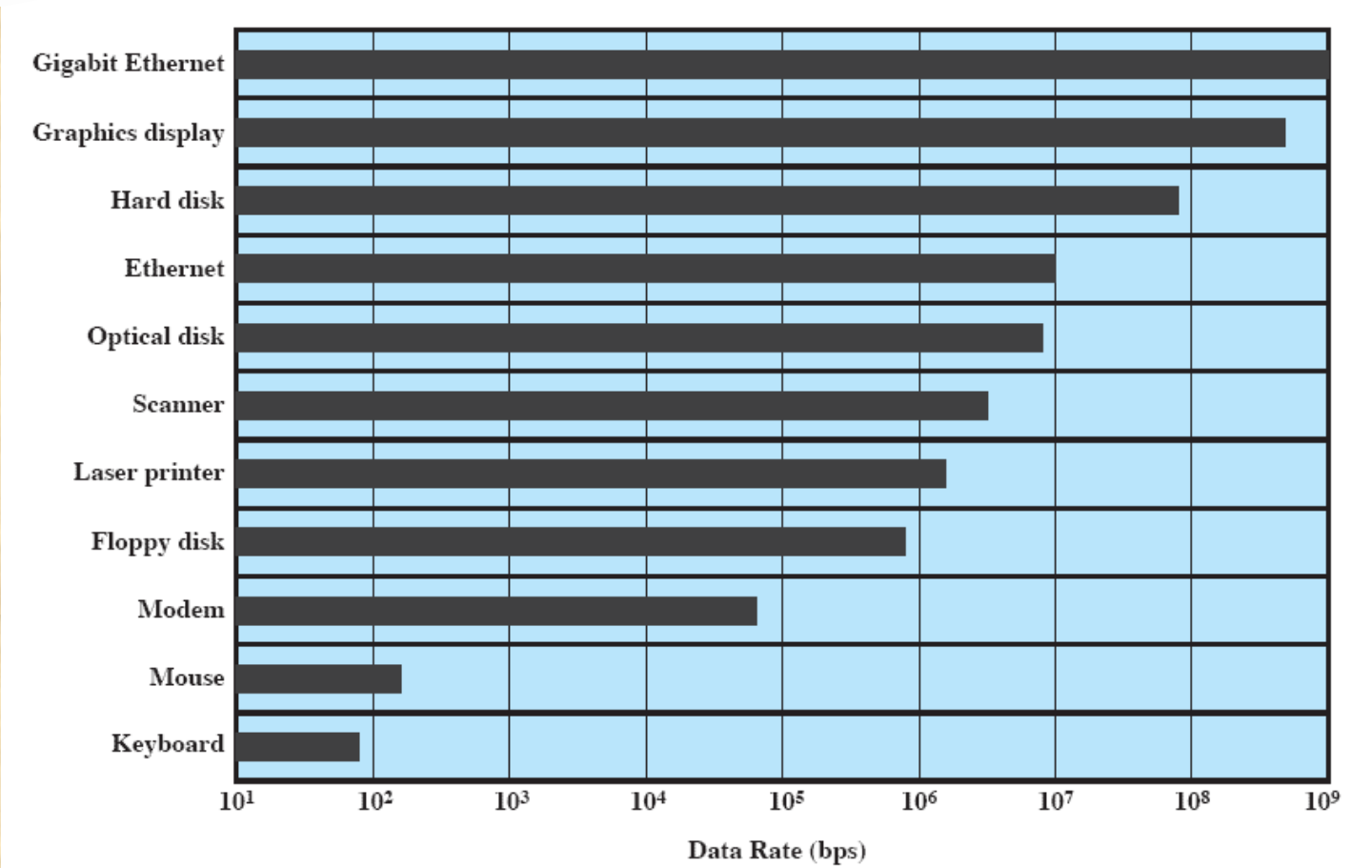


Figure 11.1 Typical I/O Device Data Rates

Organization of the I/O Function

- Three techniques for performing I/O are:
- **Programmed I/O**
  - the processor issues an I/O command on behalf of a process to an I/O module; that process then busy waits for the operation to be completed before proceeding
- **Interrupt-driven I/O**
  - the processor issues an I/O command on behalf of a process
    - if non-blocking – processor continues to execute instructions from the process that issued the I/O command
    - if blocking – the next instruction the processor executes is from the OS, which will put the current process in a blocked state and schedule another process
- **Direct Memory Access (DMA)**
  - a DMA module controls the exchange of data between main memory and an I/O module

Techniques for Performing I/O

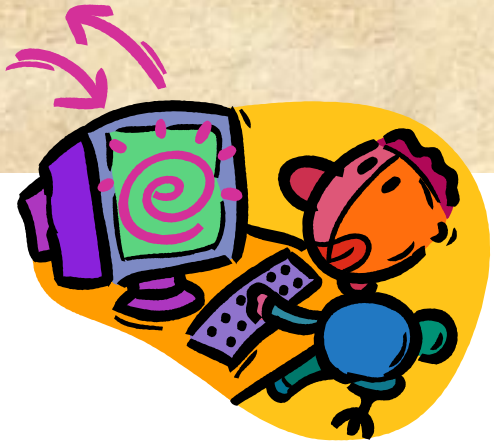


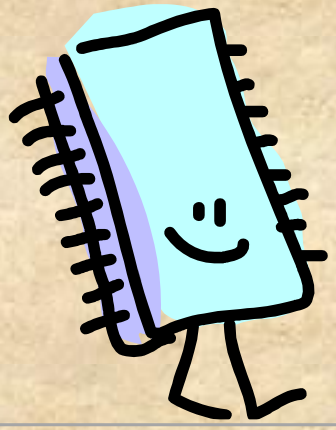
Table 11.1 I/O Techniques

	No Interrupts	Use of Interrupts
I/O-to-memory transfer through processor	Programmed I/O	Interrupt-driven I/O
Direct I/O-to-memory transfer		Direct memory access (DMA)

Evolution of the I/O Function

1. Processor directly controls a peripheral device
2. A controller or I/O module is added
3. Same configuration as step 2, but now interrupts are employed
4. The I/O module is given direct control of memory via DMA
5. The I/O module is enhanced to become a separate processor, with a specialized instruction set tailored for I/O
6. The I/O module has a local memory of its own and is, in fact, a computer in its own right

Direct Memory Access



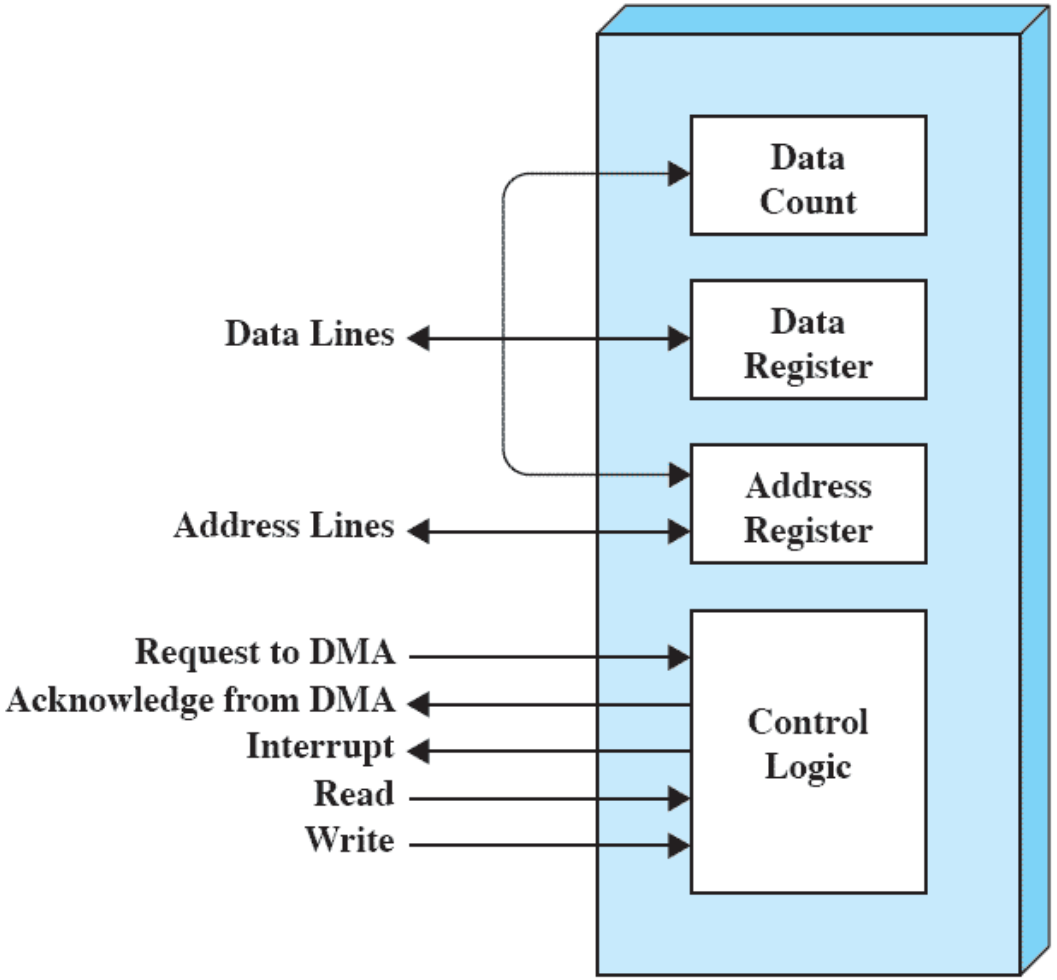


Figure 11.2 Typical DMA Block Diagram

Disk Performance Parameters

- The actual details of disk I/O operation depend on the:
  - computer system
  - operating system
  - nature of the I/O channel and disk controller hardware

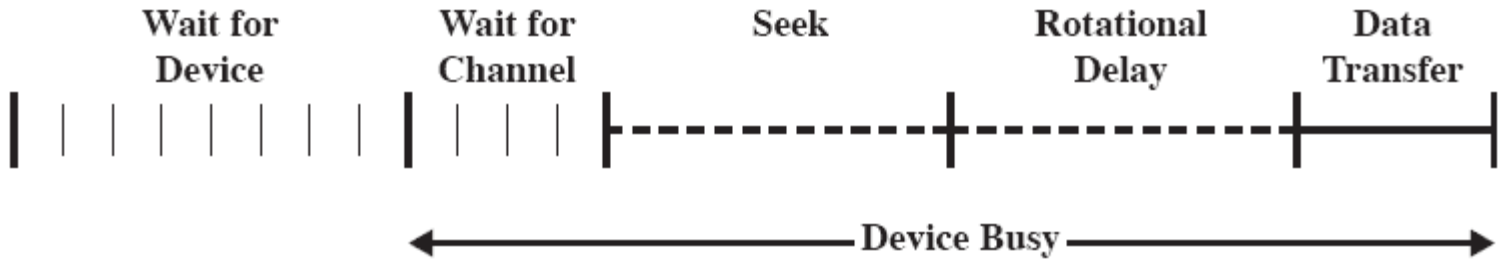


Figure 11.6 Timing of a Disk I/O Transfer

Positioning the Read/Write Heads

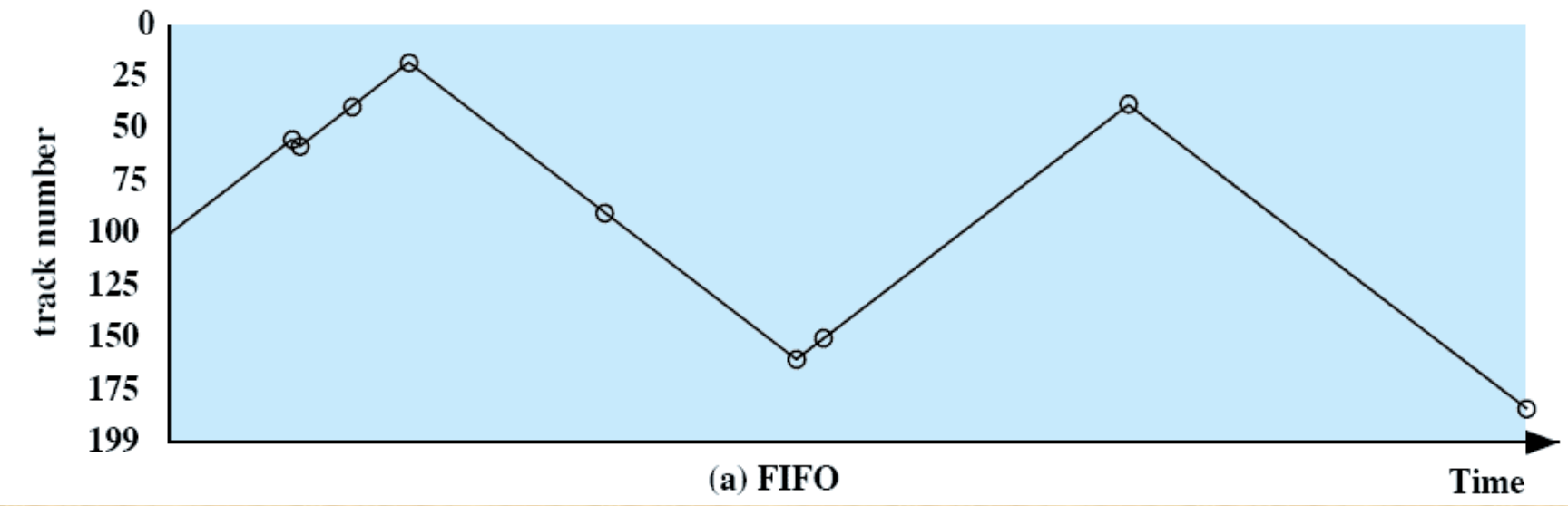
- When the disk drive is operating, the disk is rotating at constant speed
- To read or write the head must be positioned at the desired track and at the beginning of the desired sector on that track
- Track selection involves moving the head in a movable-head system or electronically selecting one head on a fixed-head system
- On a movable-head system the time it takes to position the head at the track is known as **seek time**
- The time it takes for the beginning of the sector to reach the head is known as **rotational delay**
- The sum of the seek time and the rotational delay equals the **access time**

Name	Description	Remarks
Selection according to requestor		
RSS	Random scheduling	For analysis and simulation
FIFO	First in first out	Fairest of them all
PRI	Priority by process	Control outside of disk queue management
LIFO	Last in first out	Maximize locality and resource utilization
Selection according to requested item		
SSTF	Shortest service time first	High utilization, small queues
SCAN	Back and forth over disk	Better service distribution
C-SCAN	One way with fast return	Lower service variability
N-step-SCAN	SCAN of N records at a time	Service guarantee
FSCAN	N-step-SCAN with N = queue size at beginning of SCAN cycle	Load sensitive

Table 11.3 Disk Scheduling Algorithms

First-In, First-Out (FIFO)

- Processes in sequential order
- Fair to all processes
- Approximates random scheduling in performance if there are many processes competing for the disk

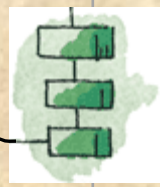


Priority (PRI)

- Control of the scheduling is outside the control of disk management software
- Goal is not to optimize disk utilization but to meet other objectives
- Short batch jobs and interactive jobs are given higher priority
- Provides good interactive response time
- Longer jobs may have to wait an excessively long time
- A poor policy for database systems

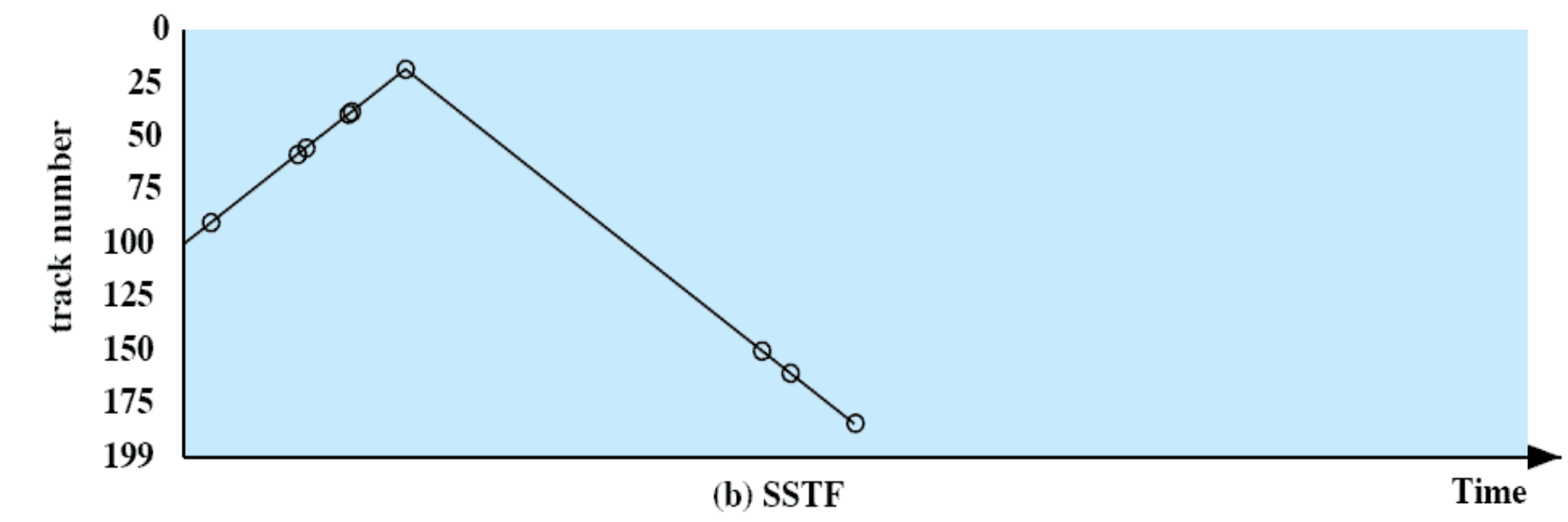
Scheduling Criteria

- Disk scheduling strategies are generally designed to treat all requests as if they have equal priority. The objective of the strategy is to optimize one or more of the following quantities:
- Throughput: number of requests processed per unit of time
- Average response time: waiting for a request to be processed
- Variance of response times: no starvation (indefinite postponement). Each request should be processed within a reasonable time period.
  - Measures fairness and predictability.



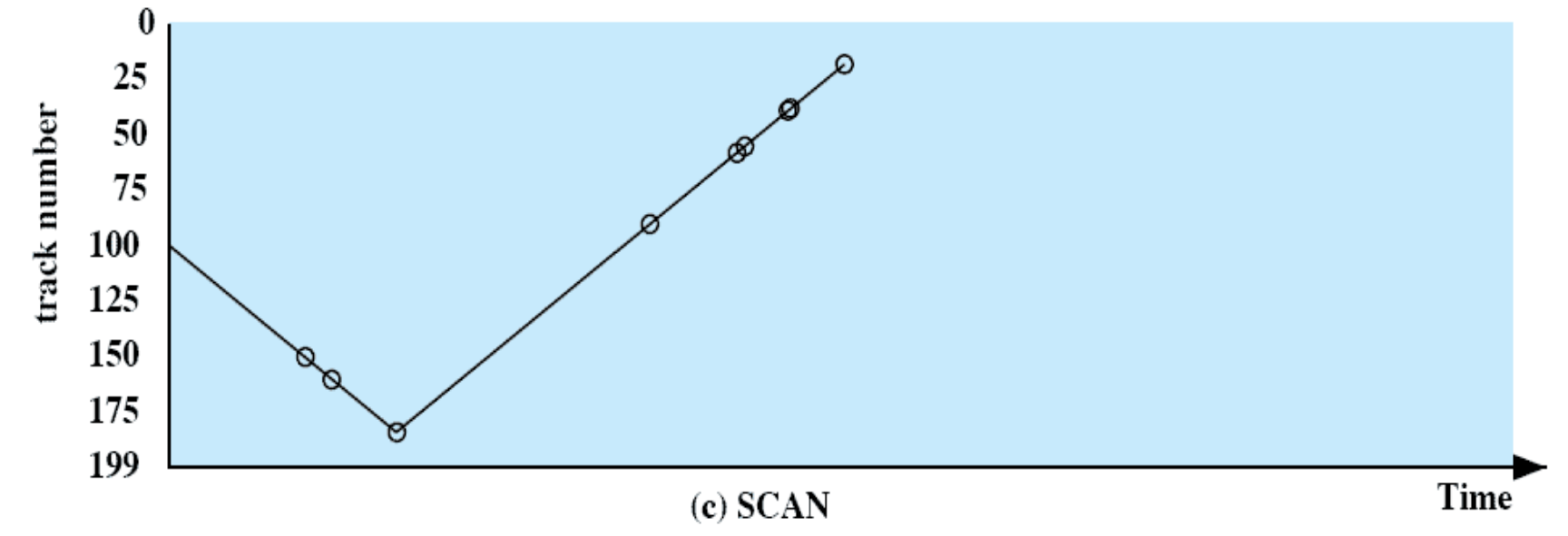
Shortest Service Time First (SSTF)

- Select the request that requires the least movement of the disk arm from its current position
- Always choose the minimum seek time
- Possible starvation: suppose requests constantly arrive for tracks between tracks 100 & 1



SCAN

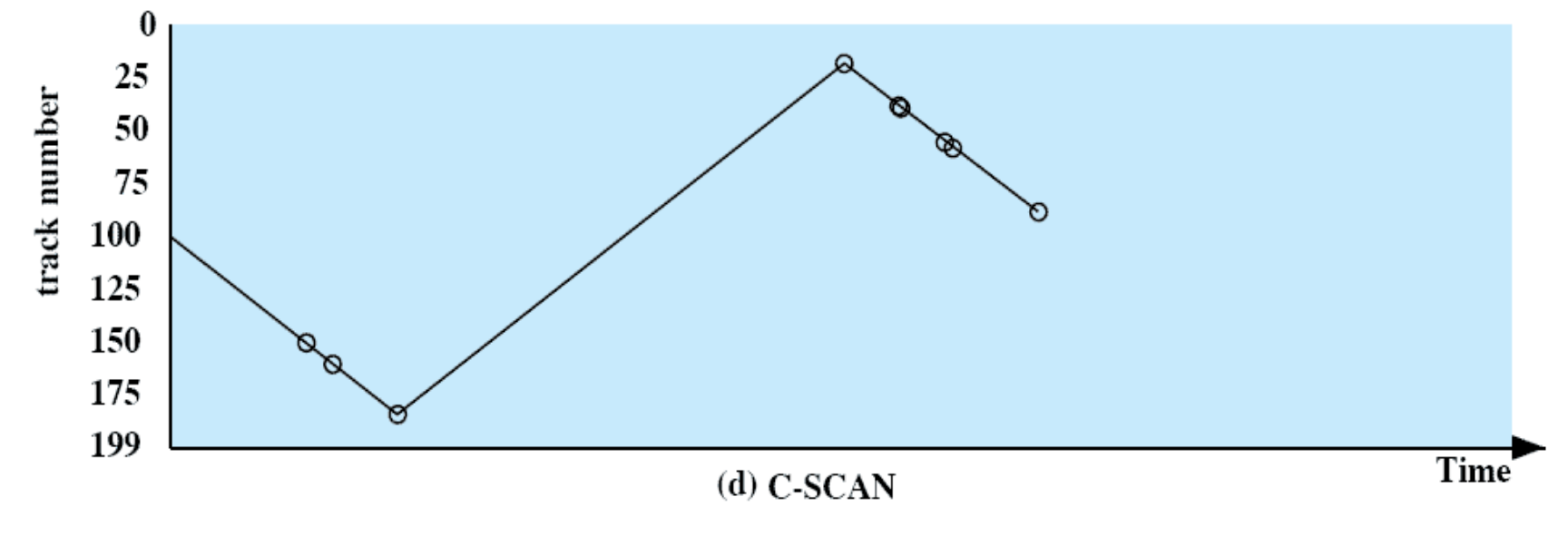
- Also known as the elevator algorithm
- Arm moves in one direction only
  - satisfies all outstanding requests until it reaches the last track in that direction then the direction is reversed
- Favors jobs whose requests are for tracks nearest to both innermost and outermost tracks but does not cause starvation.



(c) SCAN

C-SCAN (Circular SCAN)

- Restricts scanning to one direction only
- When the last track has been visited in one direction, the arm is returned to the opposite end of the disk and the scan begins again
- Reduces maximum delay for new requests.



(a) FIFO (starting at track 100)		(b) SSTF (starting at track 100)		(c) SCAN (starting at track 100, in the direction of increasing track number)		(d) C-SCAN (starting at track 100, in the direction of increasing track number)	
Next track accessed	**Number of tracks traversed**	**Next track accessed**	**Number of tracks traversed**	**Next track accessed**	**Number of tracks traversed**	**Next track accessed**	**Number of tracks traversed**
55	45	90	10	150	50	150	50
58	3	58	32	160	10	160	10
39	19	55	3	184	24	184	24
18	21	39	16	90	94	18	166
90	72	38	1	58	32	38	20
160	70	18	20	55	3	39	1
150	10	150	132	39	16	55	16
38	112	160	10	38	1	58	3
184	146	184	24	18	20	90	32
Average seek length	55.3	**Average seek length**	27.5	**Average seek length**	27.8	**Average seek length**	35.8

Table 11.2 Comparison of Disk Scheduling Algorithms

RAID

- Redundant Array of Independent Disks
- Objective: improve performance & reliability of disk I/O
- Consists of seven levels, zero through six
  - Levels 0 & 1 don't include parity checks
  - Details for striping and parity differ from level to level

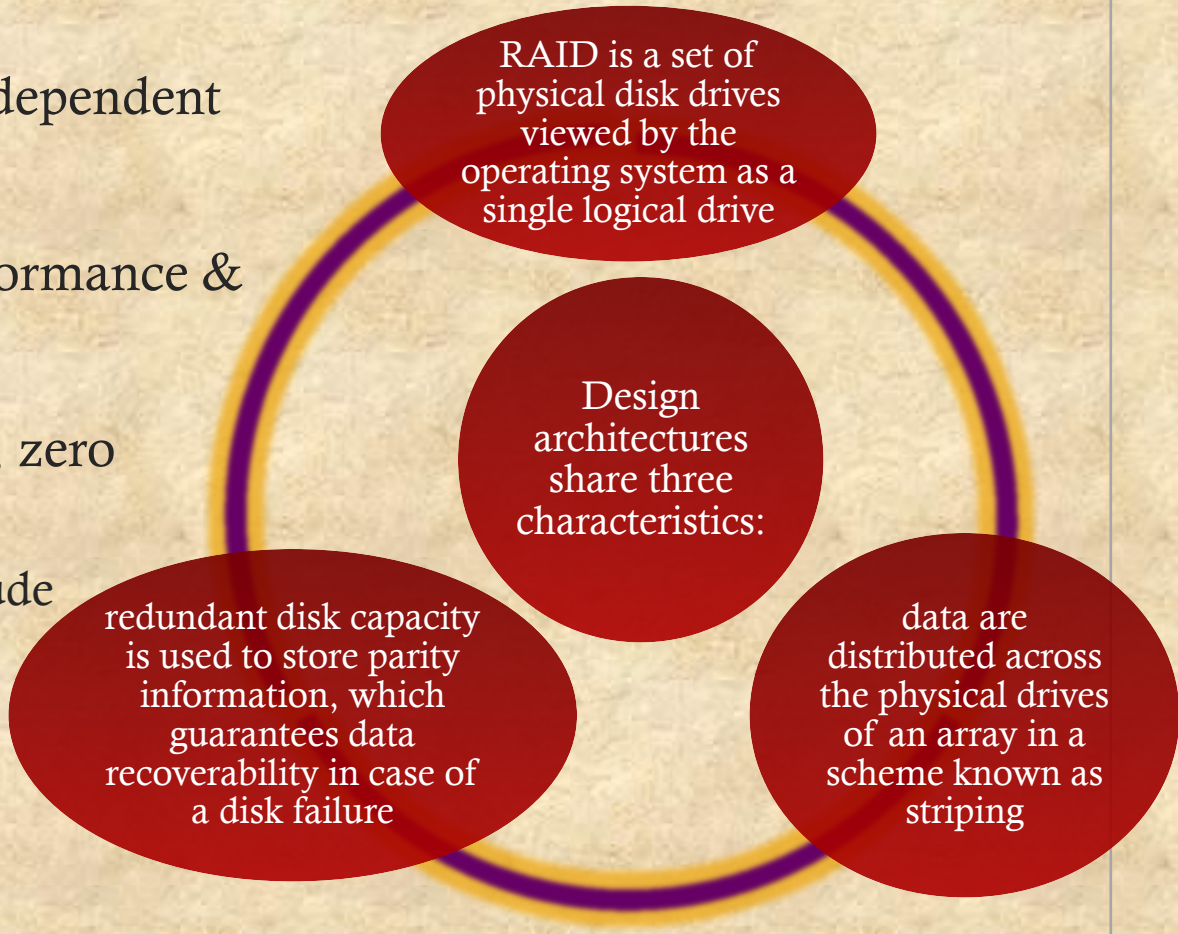


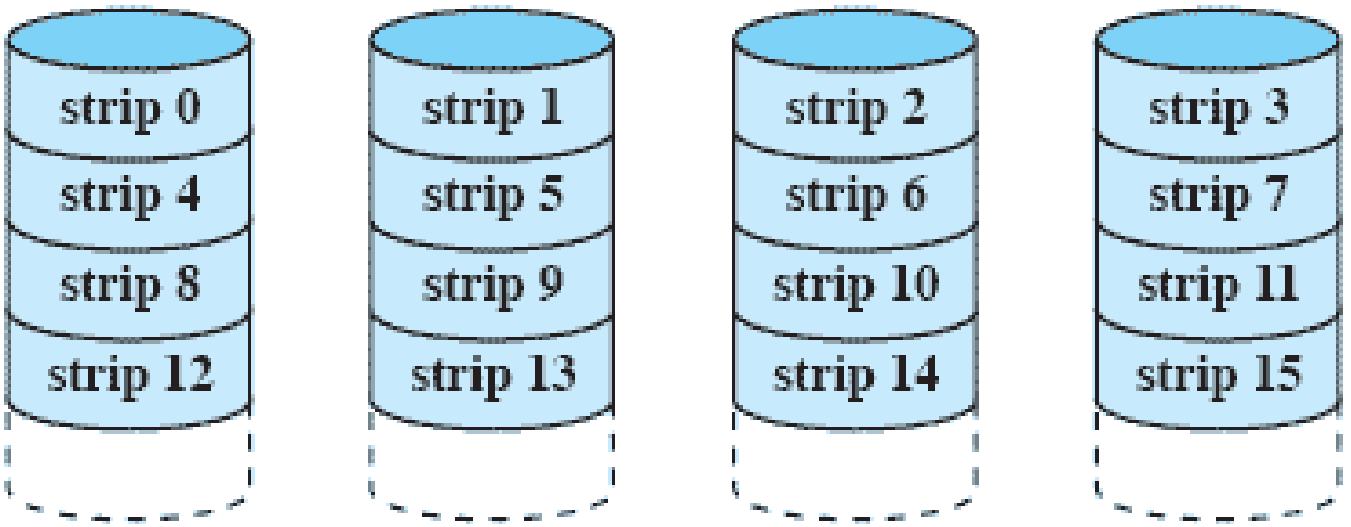
Table 11.4 RAID Levels

Category	Level	Description	Disks required	Data availability	Large I/O data transfer capacity	Small I/O request rate
Striping	0	Nonredundant	N	Lower than single disk	Very high	Very high for both read and write
Mirroring	1	Mirrored	$2N$	Higher than RAID 2, 3, 4, or 5; lower than RAID 6	Higher than single disk for read; similar to single disk for write	Up to twice that of a single disk for read; similar to single disk for write
Parallel access	2	Redundant via Hamming code	$N + m$	Much higher than single disk; comparable to RAID 3, 4, or 5	Highest of all listed alternatives	Approximately twice that of a single disk
	3	Bit-interleaved parity	$N + 1$	Much higher than single disk; comparable to RAID 2, 4, or 5	Highest of all listed alternatives	Approximately twice that of a single disk
Independent access	4	Block-interleaved parity	$N + 1$	Much higher than single disk; comparable to RAID 2, 3, or 5	Similar to RAID 0 for read; significantly lower than single disk for write	Similar to RAID 0 for read; significantly lower than single disk for write
	5	Block-interleaved distributed parity	$N + 1$	Much higher than single disk; comparable to RAID 2, 3, or 4	Similar to RAID 0 for read; lower than single disk for write	Similar to RAID 0 for read; generally lower than single disk for write
	6	Block-interleaved dual distributed parity	$N + 2$	Highest of all listed alternatives	Similar to RAID 0 for read; lower than RAID 5 for write	Similar to RAID 0 for read; significantly lower than RAID 5 for write

N = number of data disks; m proportional to log N

RAID Level 0

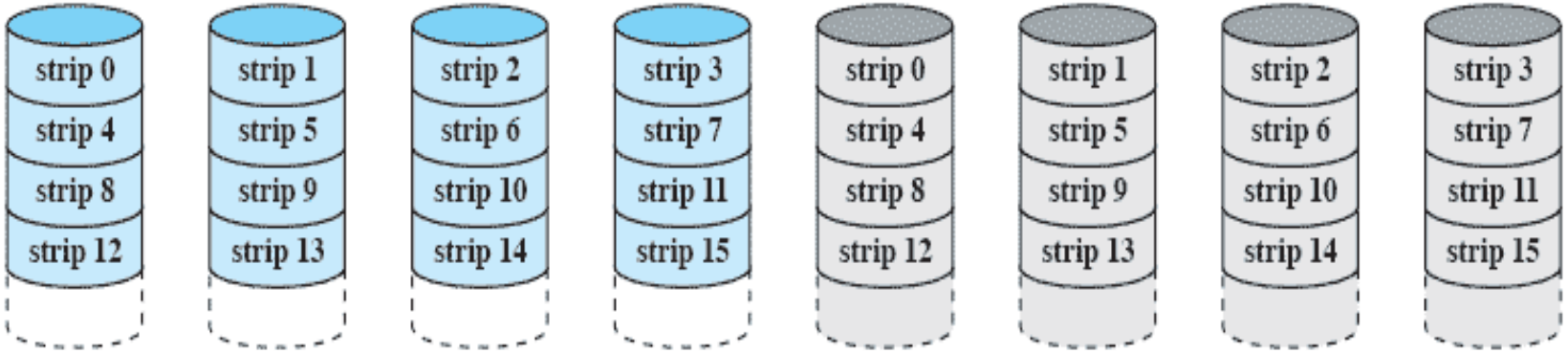
- Not a true RAID: no redundancy
- Data are distributed across all of the disks
- Two unrelated I/O requests can be done in parallel if they are on different disks, or
- Here, strips 0-3, 4-7, etc. represent a *stripe*: consecutive bytes in a file and all can be read in parallel in a single related operation.



(a) RAID 0 (non-redundant)

RAID Level 1

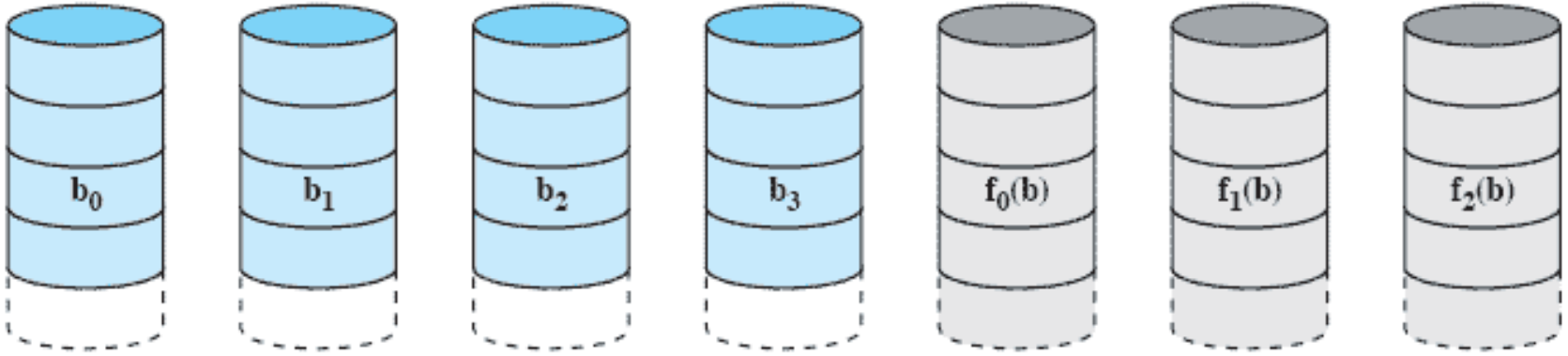
- Redundancy is achieved by the simple expedient of duplicating all the data
- There is no "write penalty" (time required to compute and update parity bits).
- When a drive fails the data may still be accessed from the second drive
- Principal disadvantage is the cost



(b) RAID 1 (mirrored)

RAID Level 2

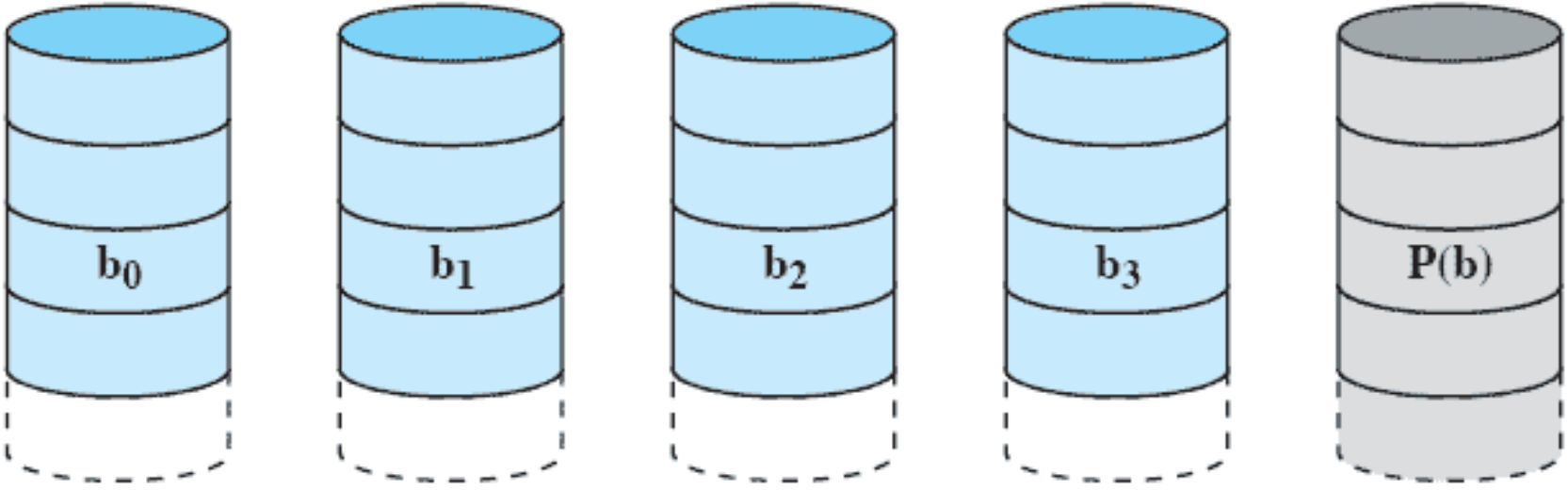
- Uses a parallel access technique: all disks involved in each operation
- Data striping is used, usually small strips
- Typically a Hamming code is used; can detect/correct single bit errors, detect double bit errors.
- Effective choice in an environment in which many disk errors occur but otherwise overkill: too many extra disks are required.



(c) RAID 2 (redundancy through Hamming code)

RAID Level 3

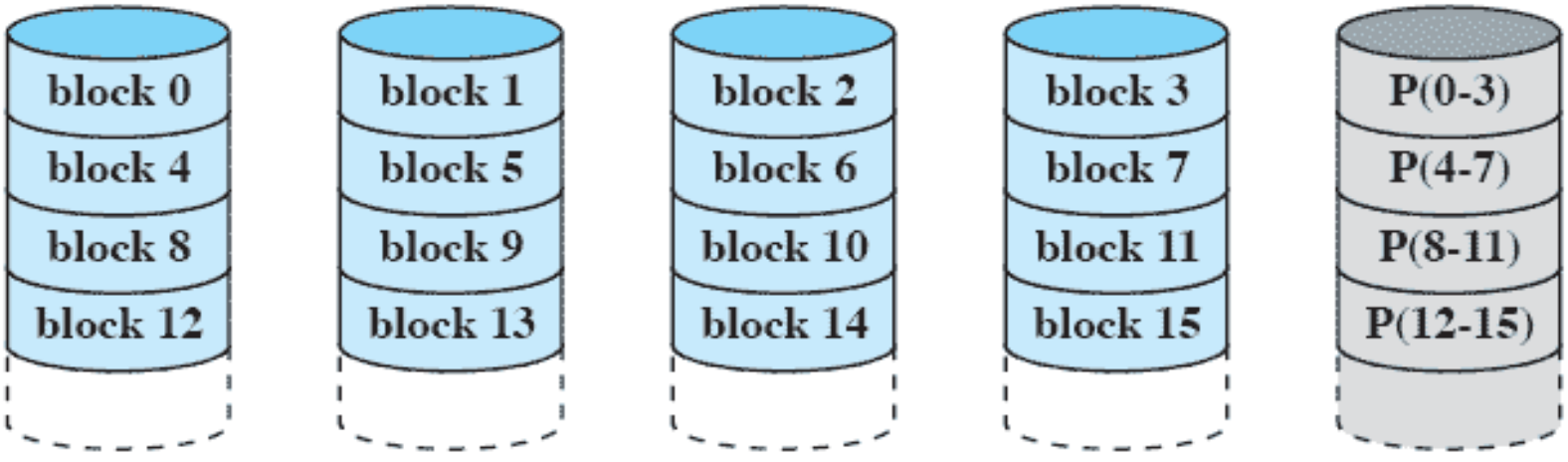
- Requires only a single redundant disk, which serves as a parity disk. If a single disk fails its contents can be reconstructed from the parity disk.
- Employs parallel access, with data distributed in small strips
- Can achieve very high data transfer rates due to small strip size.



(d) RAID 3 (bit-interleaved parity)

RAID Level 4

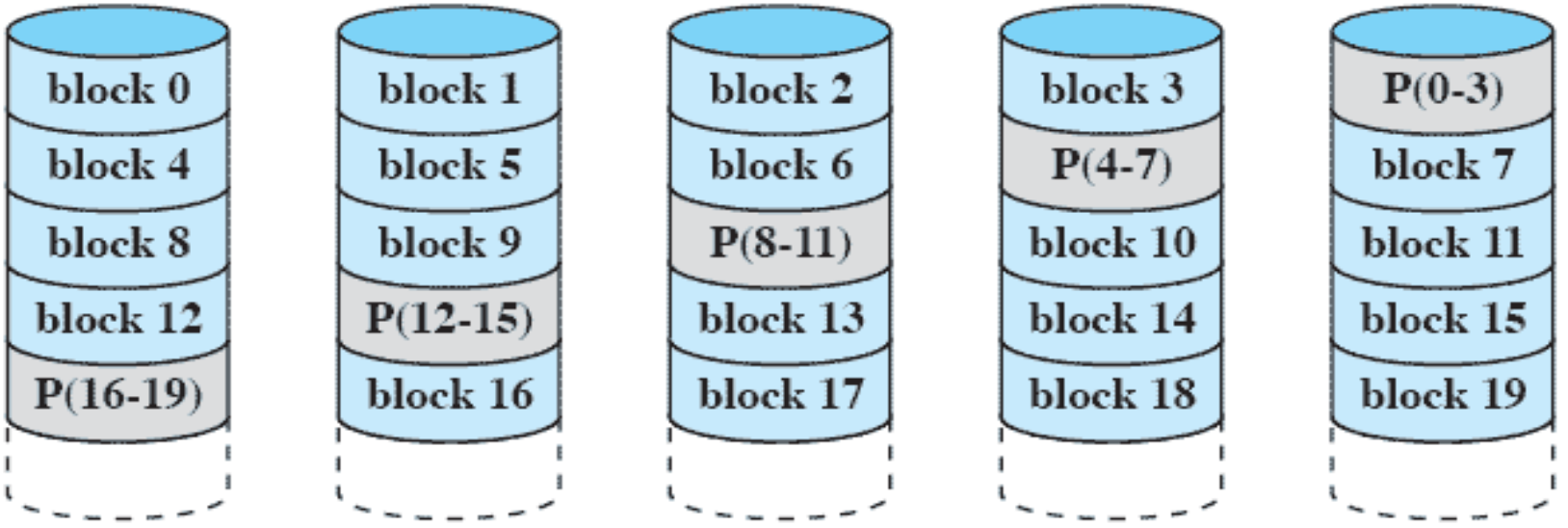
- Makes use of an independent access technique
- A bit-by-bit parity strip is calculated across corresponding strips on each data disk, and the parity bits are stored in the corresponding strip on the parity disk
- Involves a write penalty when an I/O write request of small size is performed



(e) RAID 4 (block-level parity)

RAID Level 5

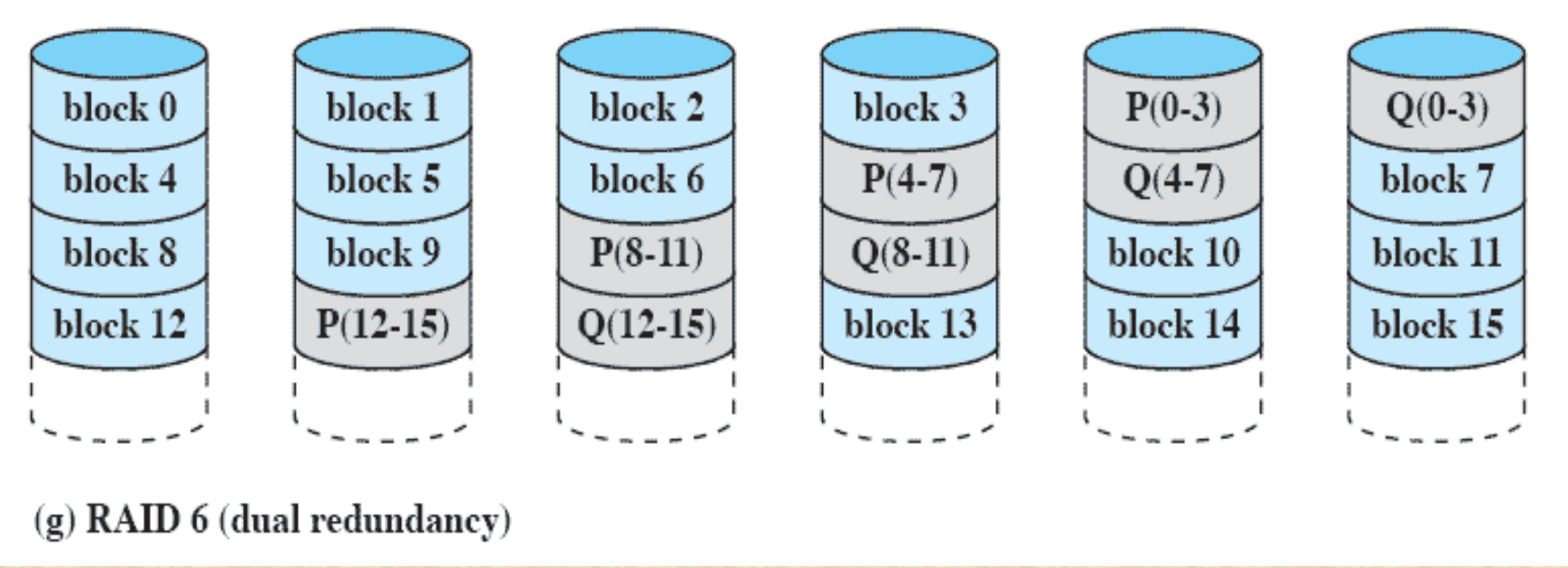
- Similar to RAID-4 but distributes the parity bits across all disks
- Typical allocation is a round-robin scheme
- Has the characteristic that the loss of any one disk does not result in data loss



(f) RAID 5 (block-level distributed parity)

RAID Level 6

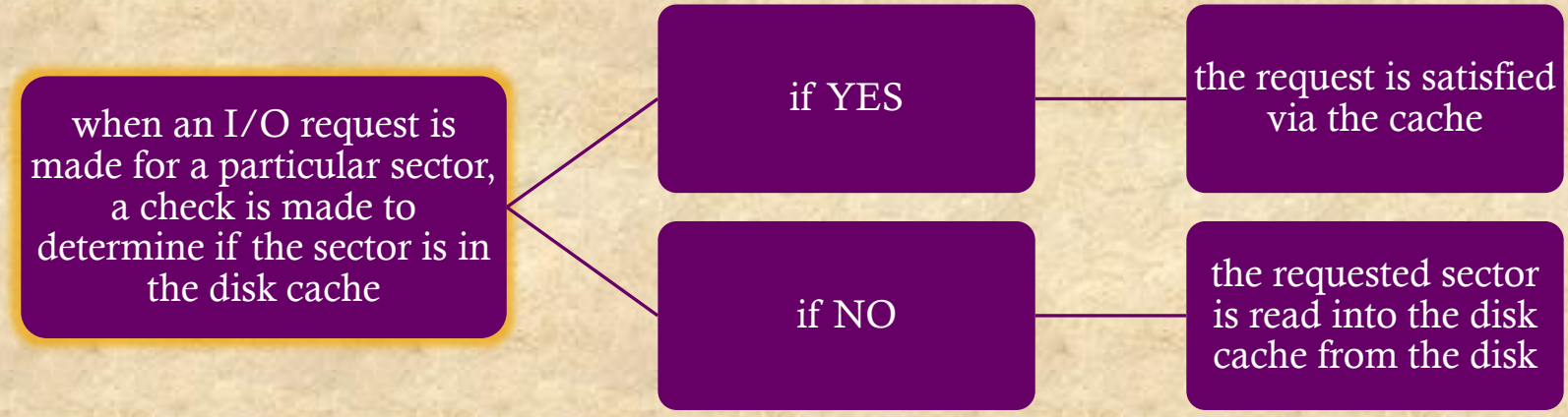
- Two different parity calculations are carried out and stored in separate blocks on different disks
- Provides extremely high data availability
- Incurs a substantial write penalty because each write affects two parity blocks



(g) RAID 6 (dual redundancy)

Disk Cache

- *Cache memory* is used to apply to a memory that is smaller and faster than main memory and that is interposed between main memory and the processor
- Reduces average memory access time by exploiting the principle of locality
- *Disk cache* is a buffer in main memory for disk sectors
- Contains a copy of some of the sectors on the disk

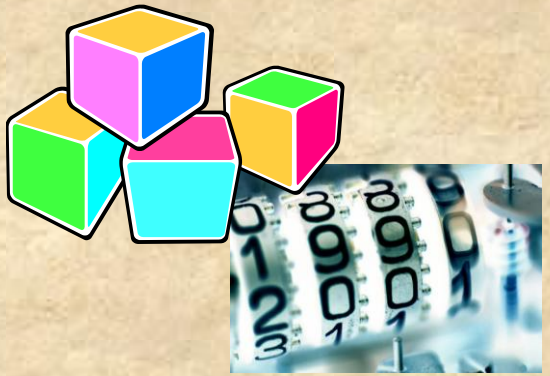


Least Recently Used (LRU)

- Most commonly used algorithm that deals with the design issue of replacement strategy
- The block that has been in the cache the longest with no reference to it is replaced
- A stack of pointers reference the cache
  - most recently referenced block is on the top of the stack
  - when a block is referenced or brought into the cache, it is placed on the top of the stack

Least Frequently Used (LFU)

- The block that has experienced the fewest references is replaced
- A counter is associated with each block
- Counter is incremented each time block is accessed
- When replacement is required, the block with the smallest count is selected



Summary

- I/O architecture is the computer system's interface to the outside world
- Disk I/O has the greatest impact on overall system performance
- Two of the most widely used approaches are disk scheduling and the disk cache
- A disk cache is a buffer, usually kept in main memory, that functions as a cache of disk block between disk memory and the rest of main memory