I/O Management and Disk Scheduling

Chapter 11, Livro do William Stallings

Sistemas de Operação, 2004-2005

Categories of I/O Devices

- Human readable
  - Printers; Display; Keyboard; Mouse.
- Machine readable
  - Disk and tap drives; Sensors; Controllers.
- Communication
  - Network drivers; Modems.

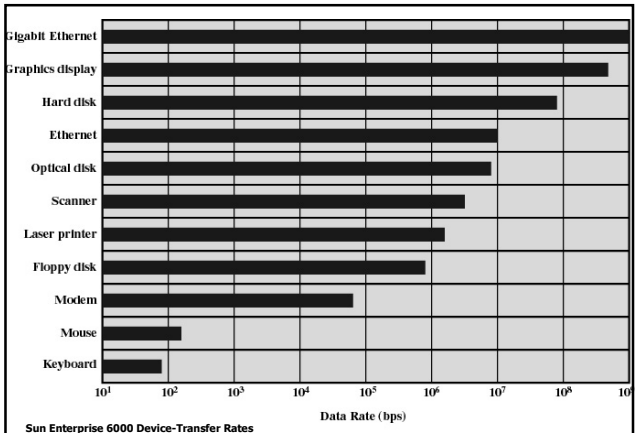
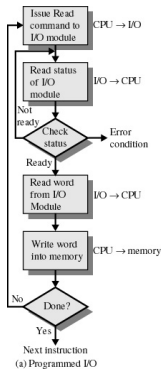


Figure 11.1 Typical I/O Device Data Rates

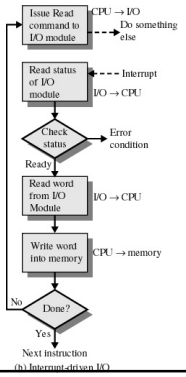
Techniques for Performing I/O

- **Programmed I/O**
  - Process is busy-waiting for the operation to complete.
  - I/O module performs the action, not the processor.
  - Sets appropriate bits in the I/O status register.
  - No interrupts occur.
  - Processor checks status until operation is complete.



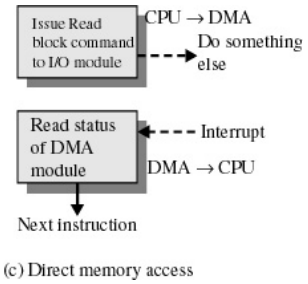
Techniques for Performing I/O

- **Interrupt-driven I/O**
  - I/O command is issued.
  - Processor continues executing instructions.
  - I/O module sends an interrupt when done.
  - No needless waiting
  - Consumes a lot of processor time because every word read or written passes through the processor



Techniques for Performing I/O

- **Direct Memory Access (DMA)**
  - DMA module controls exchange of data between main memory and the I/O device.
  - Transfers a block of data directly to or from memory.
  - An interrupt is sent when the task is complete.
  - The processor is only involved at the beginning and end of the transfer.



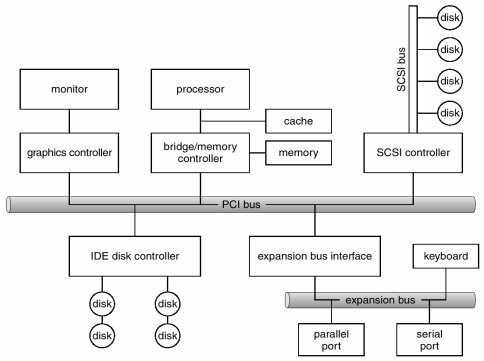
Evolution of I/O

- Processor directly controls a peripheral device.
- Controller or I/O module is added:
  - Processor uses programmed I/O without interrupts.
  - Processor does not need to handle details of external devices.
- I/O module is a separate processor, and has its local memory.

Evolution of I/O

- Controller or I/O module with interrupts:
  - Processor does not spend time waiting for an I/O operation to be performed.
- Direct Memory Access:
  - Blocks of data are moved into memory without involving the processor.
  - Processor involved at beginning and end only.

Typical PC Bus Structure



Operating System Design Issues

- **Efficiency**:
  - Most I/O devices extremely slow compared to main memory.
  - Use of multiprogramming allows for some processes to be waiting on I/O while another process executes.
  - I/O cannot keep up with processor speed.
  - Swapping is used to bring in additional Ready processes which is an I/O operation.

Operating System Design Issues

- **Generality:**
  - Desirable to handle all I/O devices in a uniform manner.
  - Hide most of the details of device I/O in lower-level routines so that processes and upper levels see devices in general terms such as `read, write, open, close, lock, unlock.`

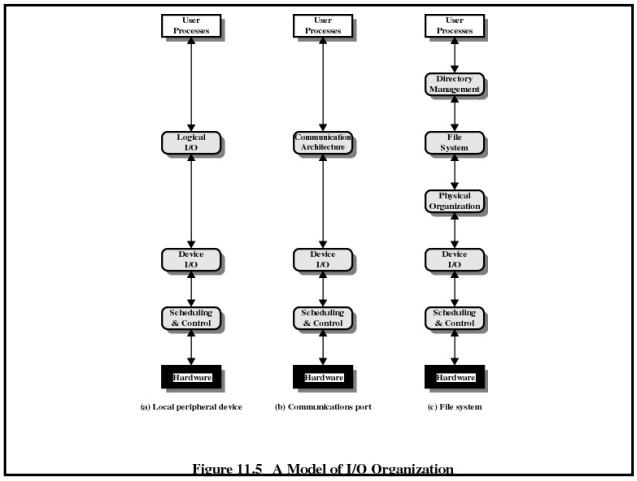


Figure 11.5 A Model of I/O Organization

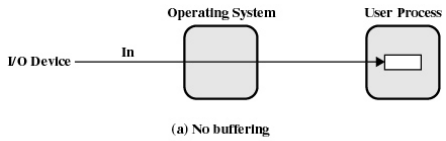
I/O Buffering

- **Reasons for buffering:**
  - Processes must wait for I/O to complete before proceeding.
  - Certain pages must remain in main memory during I/O.

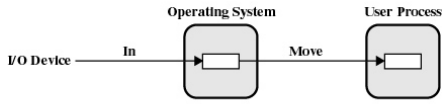
I/O Buffering

- **Block-oriented**
  - Information is stored in fixed sized blocks.
  - Transfers are made a block at a time.
  - Used for disks and tapes.
- **Stream-oriented**
  - Transfer information as a stream of bytes.
  - Used for terminals, printers, communication ports, mouse, and most other devices that are not secondary storage.

I/O Buffering



(a) No buffering



(b) Single buffering

Figure 11.6 I/O Buffering Schemes (input)

Single Buffer

- **Block-oriented:**
  - User process can process one block of data while next block is read in.
  - Swapping can occur since input is taking place in system memory, not user memory.
  - Operating system keeps track of assignment of system buffers to user processes.

Single Buffer

- **Stream-oriented:**
  - Used a line at time.
  - User input from a terminal is one line at a time with carriage return signaling the end of the line.
  - Output to the terminal is one line at a time.

I/O Buffering

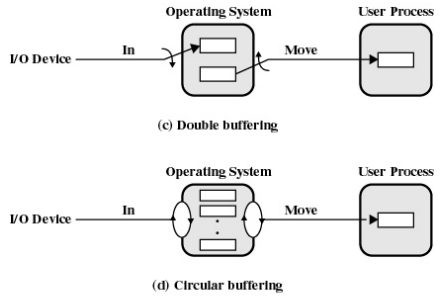


Figure 11.6 I/O Buffering Schemes (input)

Double Buffer

- Use two system buffers instead of one.
- A process can transfer data to or from one buffer while the operating system empties or fills the other buffer.

Circular Buffer

- More than two buffers are used.
- Each individual buffer is one unit in a circular buffer.
- Used when I/O operation must keep up with process.

Disk Performance Parameters

- To read or write, the disk head must be positioned at the desired track and at the beginning of the desired sector.
- **Seek time**
  - time it takes to position the head at the desired track.
- **Rotational delay or rotational latency**
  - time its takes for the beginning of the sector to reach the head.

Timing of a Disk I/O Transfer

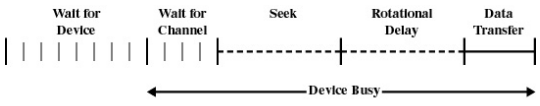


Figure 11.7 Timing of a Disk I/O Transfer

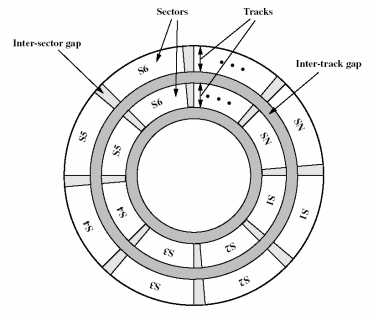
Disk Performance Parameters

- **Access time**
  - **Ta = (seek time) + (rotational delay)**
  - The time it takes to get in position to read or write.
- Data transfer occurs as the sector moves under the head.

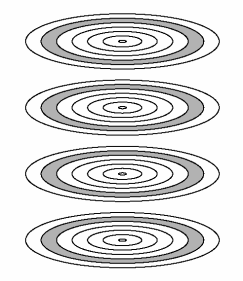
Seek Time

- Some years ago a disk was 36 cm in diameter.
- Today, the common size is about 9cm.
- Average seek time: 5 to 10 ms.

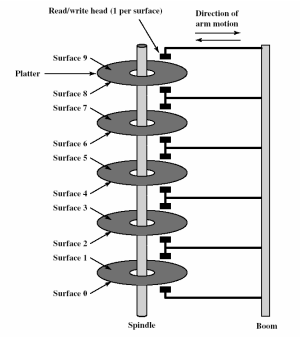
Disk Data Layout



Tracks and Cilinders



Components of a Disk Drive



Typical Disk-Drive Parameters

Characteristics	Seagate Cheetah 36	Western Digital Enterprise WDE18300
Capacity	36.4 GB	18.3 GB
Minimum track-to-track seek time	0.6 ms	0.6 ms
Average seek time	6 ms	5.2 ms
Spindle speed	10000 rpm	10000 rpm
Average rotational delay	3 ms	3 ms
Maximum transfer rate	313 Mbps	360 Mbps
Bytes per sector	512	512
Sectors per track	300	320
Tracks per cylinder (number of platter surfaces)	24	8
Cylinders (number of tracks on one side of platter)	9801	13614

Rotational Delay

- Magnetic disks have rotational speeds in the range: 5400 – 10,000 rpm.
- At 10,000 rpm the rotational delay is 3ms.
- Floppy disks rotate between 300 and 600 rpm.
- The average delay will be between 100 and 200 ms.

Transfer Time

- $$T = \frac{b}{r \cdot N}$$

T = transfer time

b = number of bytes to be transferred

N = number of bytes on a track

r = rotation speed, in revolutions per second

Average Total Access Time

- $$Ta = Ts + \frac{1}{2r} + \frac{b}{rN}$$

Ts = average seek time

Example

- Consider a disk with average seek time of 10 ms, rotation speed of 10,000 rpm, 512-byte sectors with 320 sectors per track.
- Suppose we want to read a file with 2560 sectors for a total of 1.3MBytes.
- What is the **Total Transfer Time?**

Example (v1)

- First, we assume that the file is stored on contiguous on disk, and occupies all the sectors in 8 adjacent tracks (8 tracks x 320 sectors/track = 2560 sectors).
- This corresponds to a sequential storage of the file.
- T1 = Time to read first track:
  - T1 = 10 (seek time) + 3 (rotational delay) + 6 (read 320 sectors) = 19ms
  - Next reads the seek time will be 0.
  - Each sucessive track is read in (3 + 6) = 9 ms
- **Total time** = 19 + 7 x 9 = 82 ms = **0,082 seconds**

Example (v2)

- Now, assume the file is spreaded over the disk, and the acessess to the sectors are distributed randomly.
- For each sector we have:
  - T sector = 10 (seek) + 3 (rotational delay) + 0.01875 (read 1 sector)
- **Total time** = 2560 x 13.01875 = 33328 ms = **33.328 seconds**

6/320

- **In this case it is 406 times slower!!!**

Conclusion from this example

- The order in which the sectors are read from the disk has a tremendous effect on I/O performance.
- Seek time is the reason for differences in performance.
- For a single disk request there will be a number of I/O requests.
- If requests are selected randomly, we will get the worst possible performance.

Disk Scheduling Policies

Disk Scheduling Policies

- **First-in, first-out (FIFO)**
  - Process request sequentially.
  - Fair to all processes.
  - Approaches random scheduling in performance if there are many processes.

Disk Scheduling Policies

- **Priority**
  - Goal is not to optimize disk use but to meet other objectives.
  - Short batch jobs may have higher priority.
  - Provide good interactive response time.

Disk Scheduling Policies

- **Last-in, first-out**
  - Good for transaction processing systems.
    - The device is given to the most recent user so there should be little arm movement
  - Possibility of starvation since a job may never regain the head of the line.

Disk Scheduling Policies

- **Shortest Service Time First (SSTF)**
  - Select the disk I/O request that requires the least movement of the disk arm from its current position.
  - Always choose the minimum Seek time.

Disk Scheduling Policies

- **SCAN**
  - 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.

LOOK and C-LOOK: variations of SCAN and C-SCAN

Disk Scheduling Policies

- **C-SCAN**
  - Restricts scanning to one direction only.
  - 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.

LOOK and C-LOOK: variations of SCAN and C-SCAN

Disk Scheduling Policies

- **N-step-SCAN**
  - Segments the disk request queue into sub-queues of length N.
  - Sub-queues are process one at a time, using SCAN.
  - New requests added to other queue when queue is processed.
- **FSCAN**
  - Two queues.
  - One queue is empty for new request.

Comparing Disk Scheduling Algorithms

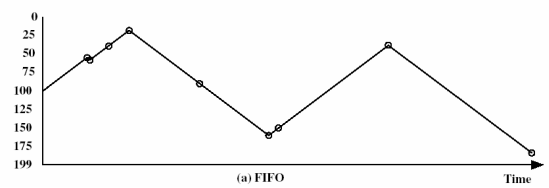
- **FIFO, SSTF, LOOK, C-LOOK**
- Example:

  55,58,39,18,90,160,150,38,184.

- Initial sector=100

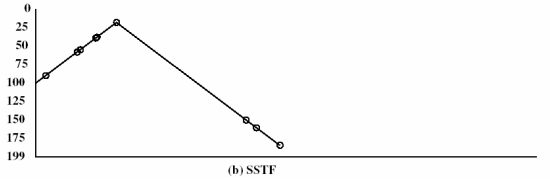
Comparing Disk Scheduling Algorithms: FIFO

55,58,39,18,90,160,150,38,184



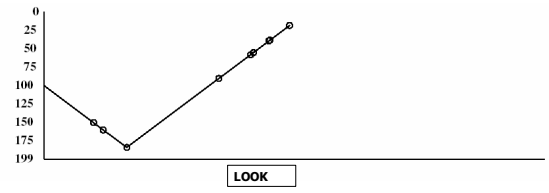
Comparing Disk Scheduling Algorithms: SSTF

55,58,39,18,90,160,150,38,184



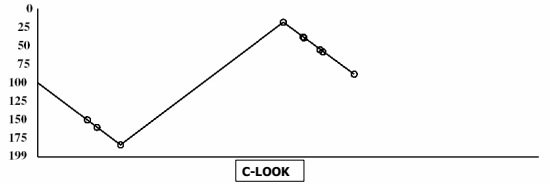
Comparing Disk Scheduling Algorithms: LOOK

55,58,39,18,90,160,150,38,184



Comparing Disk Scheduling Algorithms: C-LOOK

55,58,39,18,90,160,150,38,184



Disk Scheduling Algorithms

Table 11.3 Disk Scheduling Algorithms [WIED87]

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

Comparing Disk Scheduling Algorithms

(a) FIFO (starting at track 100)		(b) SSTF (starting at track 100)		LOOK (starting at track 100, in the direction of increasing track number)		C-LOOK (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

Disk Scheduling: Another Example

- Request queue (0-199).

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

  Head pointer 53

FCFS

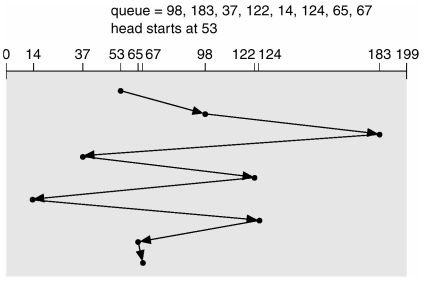


Illustration shows total head movement of 640 cylinders.

SSTF

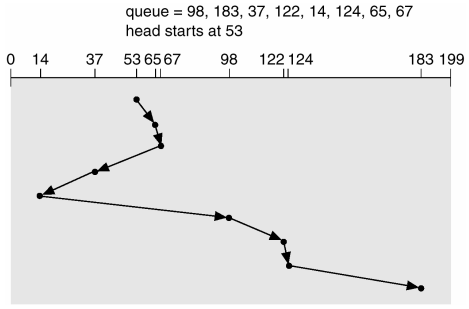
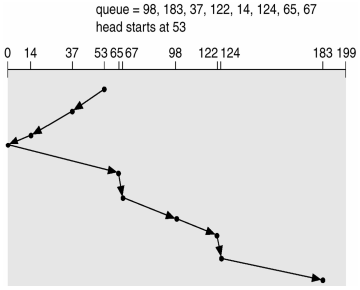


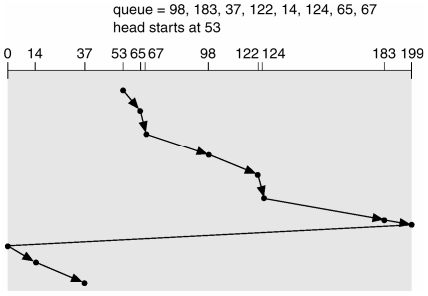
Illustration shows total head movement of 236 cylinders.

SCAN



- Illustration shows total head movement of 208 cylinders.

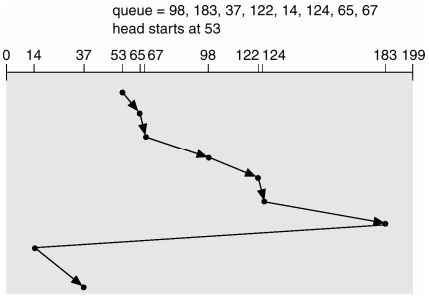
C-SCAN



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.

C-LOOK



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.

Go to QUIZ#12

Disk Reliability

MTBF: Disk Reliability

- MTBF: Mean Time Between Failures
- Example: MTBF (1 disk)=100.000 horas
- Se tivermos 100 discos...
- MTBF = 100.000/100=1.000 horas=41 dias

Mirror Disks

- Mean Time to Repair = 10 horas
- MTBF = 100.000 horas
- Pressuposto: avarias independentes.
- Mean Time to Data Loss in a Mirror Disk System
  $= 100.000^2 / (2 \times 10)$
  $= 500 \times 10^6 = 57.000$ anos...

RAID Systems

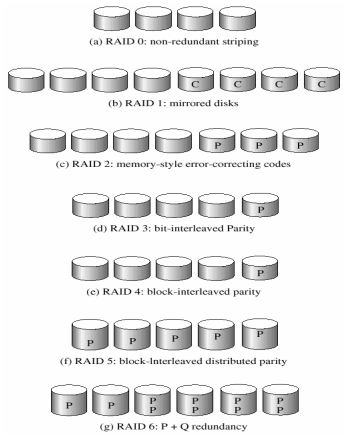
RAID Disks

- Array of Disks thats operate independently and in parallel.
- With multiple disks, we can perform separate I/O requests in parallel.
- Using data redundancy we can improve the reliability of data stored on disks.

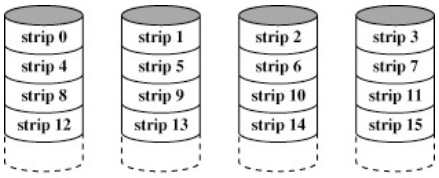
RAID Levels

Category	Level	Description	I/O Request Rate (Read/Write)	Data Transfer Rate (Read/Write)	Typical Application
Striping	0	Nonredundant	Large strips: Excellent	Small strips: Excellent	Applications requiring high performance for noncritical data
Mirroring	1	Mirrored	Good/Fair	Fair/Fair	System drives; critical files
Parallel access	2	Redundant via Hamming code	Poor	Excellent	
	3	Bit-interleaved parity	Poor	Excellent	Large I/O request size applications, such as imaging, CAD
Independent access	4	Block-interleaved parity	Excellent/Fair	Fair/Poor	
	5	Block-interleaved distributed parity	Excellent/Fair	Fair/Poor	High request rate, read-intensive, data lookup
	6	Block-interleaved dual distributed parity	Excellent/Poor	Fair/Poor	Applications requiring extremely high availablity

Levels 2 and 4 are not comercially available.



RAID 0 (non-redundant)



(a) RAID 0 (non-redundant)

Figure 11.9 RAID Levels (page 1 of 2)

RAID

- RAID is a set of physical disks drives viewed by the operating system as a single logical drive.
- Data is distributed across the physical drives of an array.
- Redundant disk capacity is used to store parity information, which guarantees data recoverability in case of a disk failure.

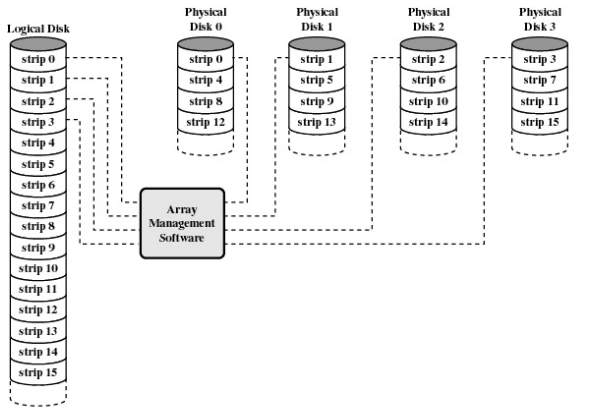
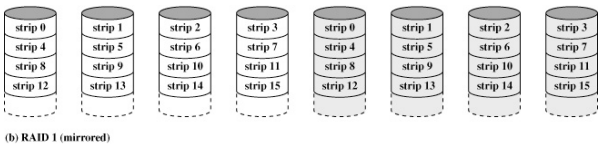


Figure 11.10 Data Mapping for a RAID Level 0 Array

RAID 1 (mirrored)

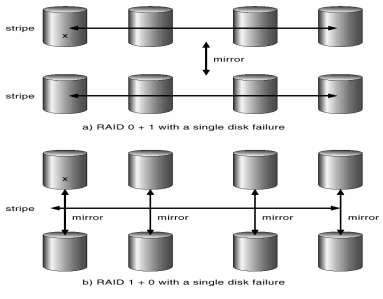


(b) RAID 1 (mirrored)

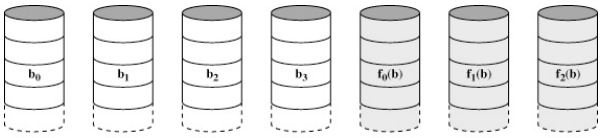
Figure 11.9 RAID Levels (page 1 of 2)

- Read requests: can be faster than in RAID-0
- Write requests: are done in parallel into two disks
- Advantage: tolerates disk failures
- Disadvantage: cost.

RAID "0+1" vs RAID "1 +0"



RAID 2 (redundancy through Hamming code)

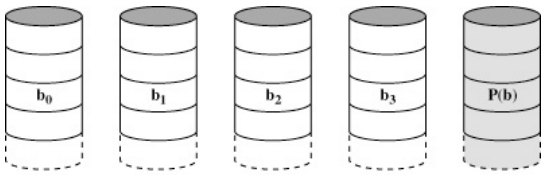


(c) RAID 2 (redundancy through Hamming code)

Figure 11.9 RAID Levels (page 1 of 2)

- An error-correcting code is calculated across corresponding bits on each data disk.
- Hamming code: is able to correct single-bit errors and detect double-bit errors.
- On a read request: all disks are simultaneously accessed.
- On a write request: all data disks and parity disks must be accessed.
- RAID-2: is costly and usually not implemented.

RAID 3 (bit-interleaved parity)

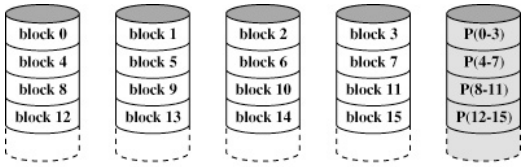


(d) RAID 3 (bit-interleaved parity)

Figure 11.9 RAID Levels (page 2 of 2)

- A parity bit is computed for the set of individual bits in the same position in the data disks.
- P4(i) = D0(i) xor D1(i) xor D2(i) xor D3(i)
- Tolerates (corrects) one-bit failure.

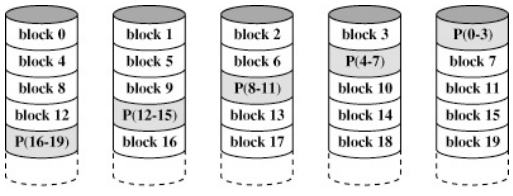
RAID 4 (block-level parity)



(e) RAID 4 (block-level parity)

Figure 11.9 RAID Levels (page 2 of 2)

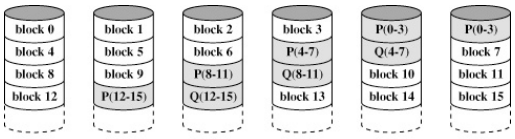
RAID 5 (block-level distributed parity)



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

Figure 11.9 RAID Levels (page 2 of 2)

RAID 6 (dual redundancy)



(g) RAID 6 (dual redundancy)

Figure 11.9 RAID Levels (page 2 of 2)

- RAID6 array: requires N+2 disks
- Uses two different data check algorithms (Q and P).
- Makes it possible to regenate data even if two disks fail.

Go to QUIZ#13

Disk Cache

Disk Cache

- Buffer in main memory for disk sectors.
- Contains a copy of some of the sectors on the disk.
- Three algorithms:
  - **Least-Recently Used (LRU)**
  - **Least-Frequently Used (LFU)**
  - **Frequency-based Replacement.**

Least Recently Used

- The block that has been in the cache the longest with no reference to it is replaced.
- The cache consists of a stack of blocks.
- 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.

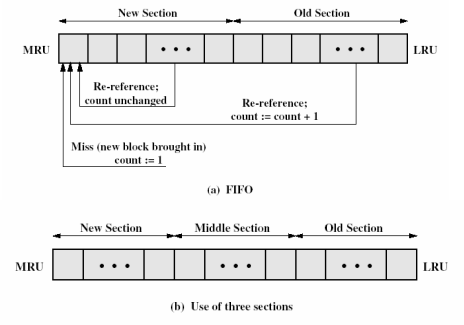
Least Recently Used

- The block on the bottom of the stack is removed when a new block is brought in.
- Blocks don't actually move around in main memory.
- A stack of pointers is used.

Least Frequently Used

- The block that has experienced the fewest references is replaced.
- A counter is associated with each block.
- Counter is incremented each time block accessed.
- Block with smallest count is selected for replacement.
- Some blocks may be referenced many times in a short period of time and then not needed any more.
- Optimize with: **Frequency-based Replacement** strategy.

Frequency-based Replacement



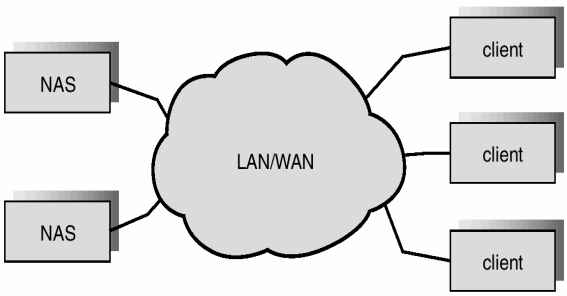
Remote Disks

Disk Attachment

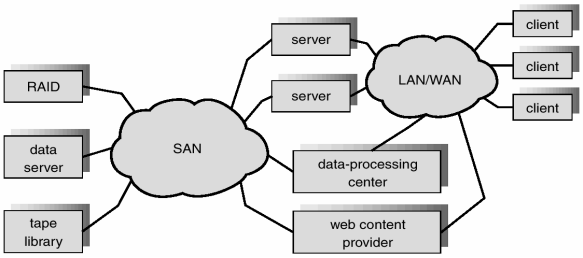
- Disks may be attached one of two ways:

1. **Host attached** via an I/O port
2. **Network attached** via a network connection

Network-Attached Storage



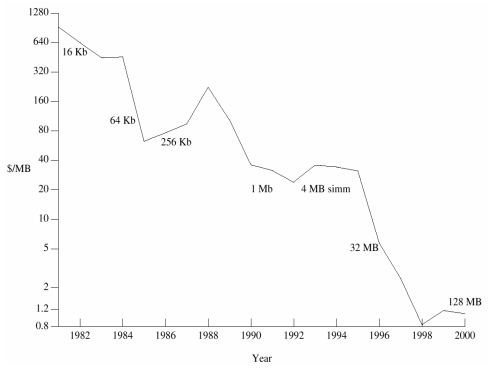
Storage-Area Network



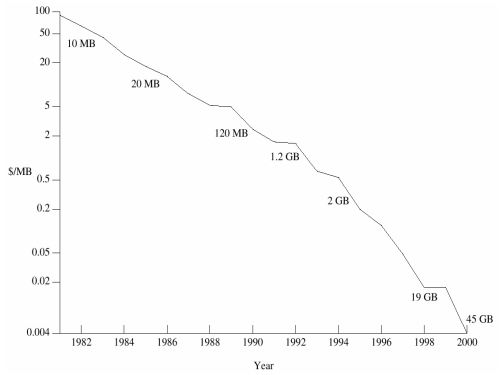
Cost

- 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 2000



Price per Megabyte of Magnetic Hard Disk, From 1981 to 2000



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