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

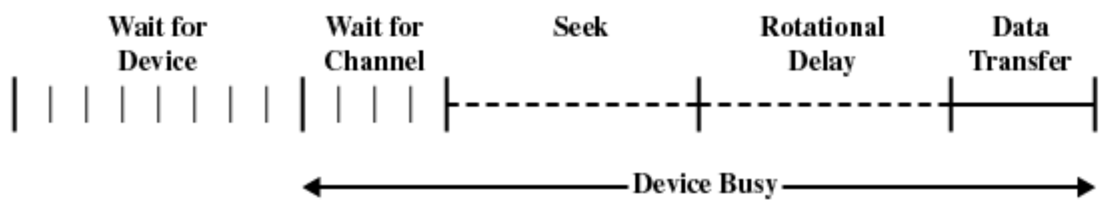


Figure 11.6 Timing of a Disk I/O Transfer

Disk Performance Parameters

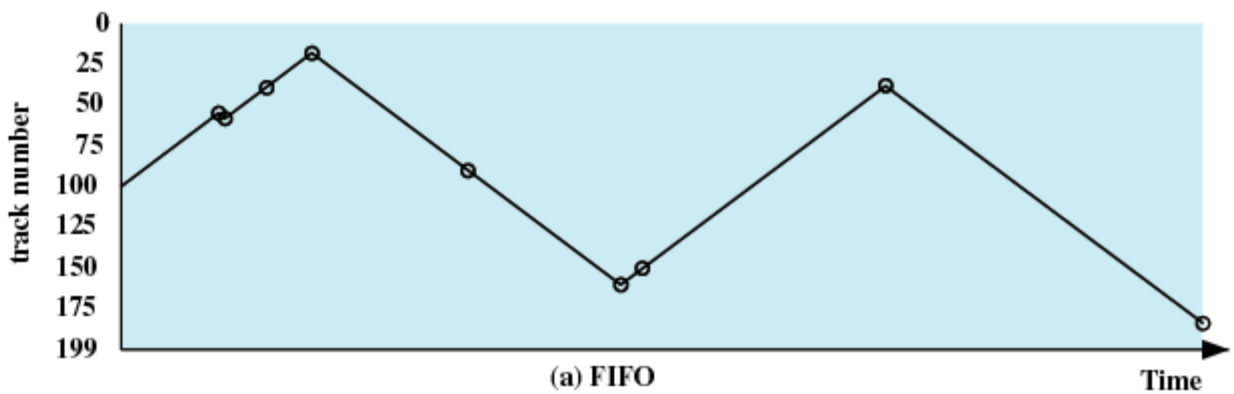
- Access time
  - Sum of seek time and rotational delay
  - The time it takes to get in position to read or write
- Data transfer occurs as the sector moves under the head

Disk Scheduling Policies

- Seek time is the reason for differences in performance
- For a single disk there will be a number of I/O requests
- If requests are selected randomly, we will poor performance

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



(a) FIFO

Disk Scheduling Policies

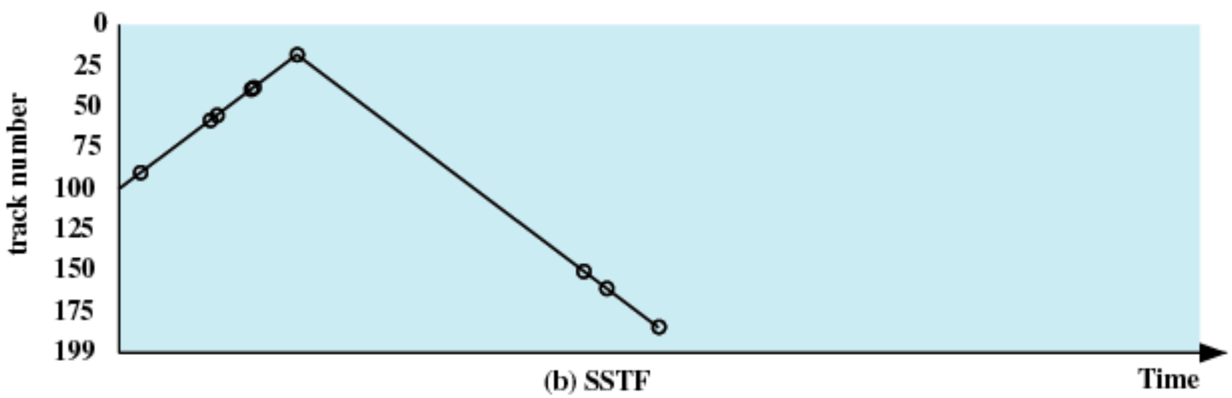
- Priority
  - Goal is not to optimize disk use but to meet other objectives
  - Short batch jobs may have higher priority
  - Provide good interactive response time

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

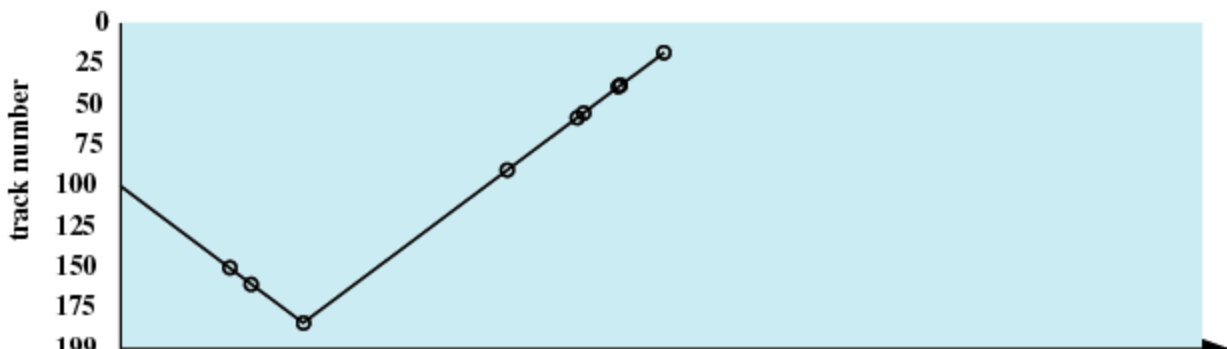
- Shortest Service Time First
  - Select the disk I/O request that requires the least movement of the disk arm from its current position
  - Always choose the minimum Seek time



(b) SSTF

Disk Scheduling Policies

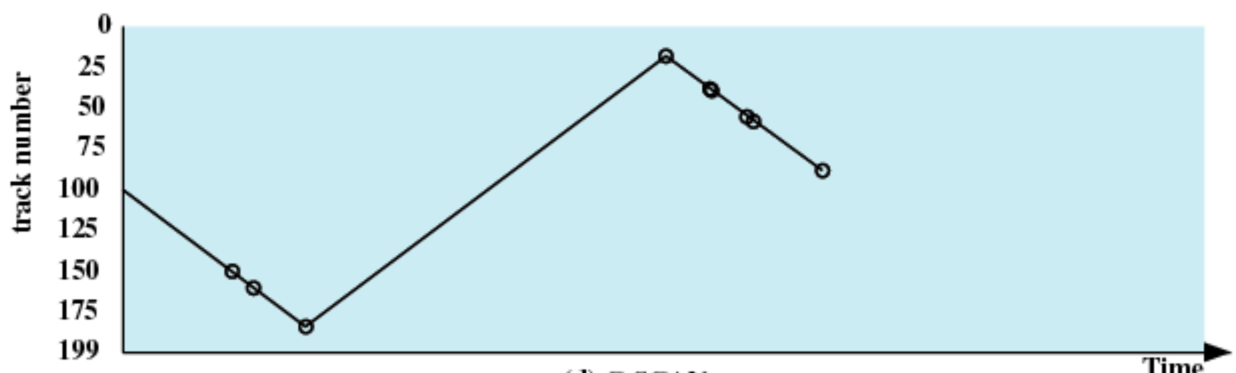
- SCAN
  - Arm moves in one direction only, satisfying all outstanding requests until it reaches the last track in that direction
  - Direction is reversed



(c) SCAN

Disk Scheduling Policies

- C-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



(d) C-SCAN

Disk Scheduling Policies

- N-step-SCAN
  - Segments the disk request queue into subqueues of length N
  - Subqueues are processed 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 requests

Disk Scheduling Algorithms

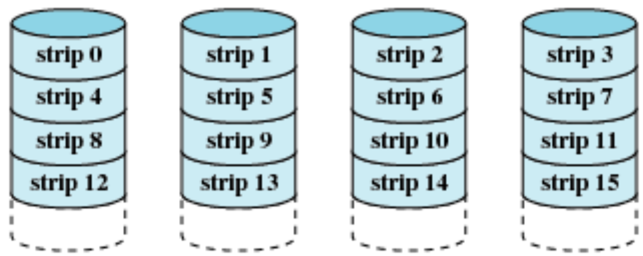
Table 11.2 Comparison of Disk Scheduling Algorithms

(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

RAID

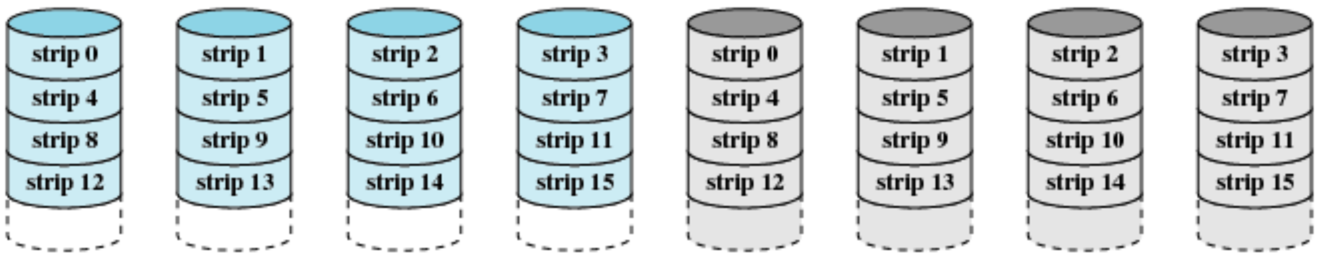
- Redundant Array of Independent Disks
- Set of physical disk drives viewed by the operating system as a single logical drive
- Data are distributed across the physical drives of an array
- Redundant disk capacity is used to store parity information

RAID 0 (non-redundant)



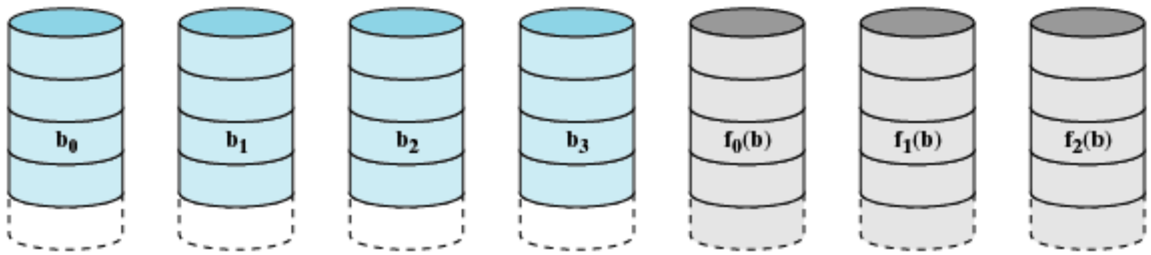
(a) RAID 0 (non-redundant)

RAID 1 (mirrored)



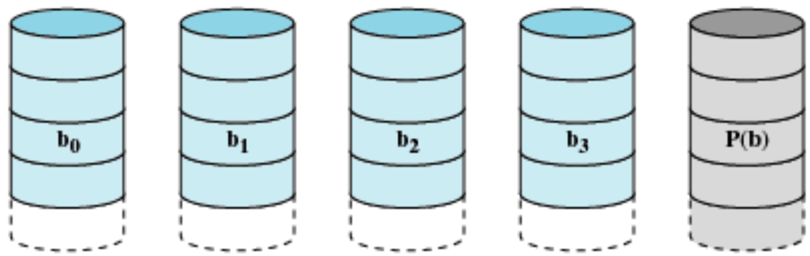
(b) RAID 1 (mirrored)

RAID 2 (redundancy through Hamming code)



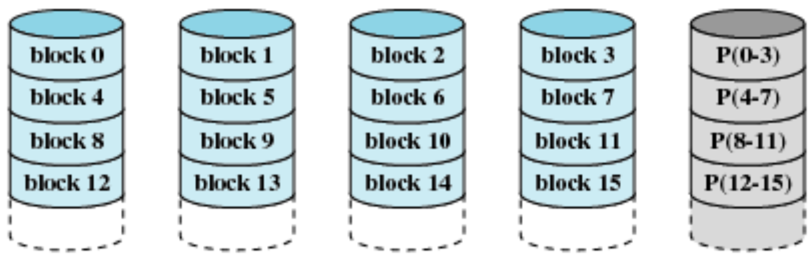
(c) RAID 2 (redundancy through Hamming code)

RAID 3 (bit-interleaved parity)



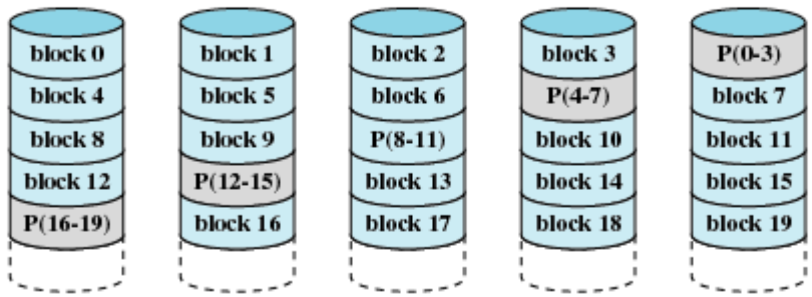
(d) RAID 3 (bit-interleaved parity)

RAID 4 (block-level parity)



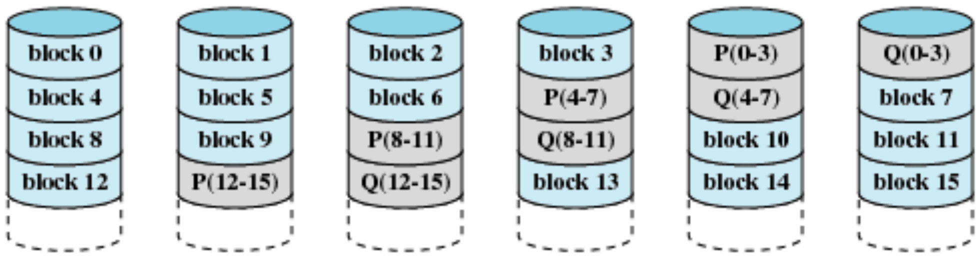
(e) RAID 4 (block-level parity)

RAID 5 (block-level distributed parity)



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

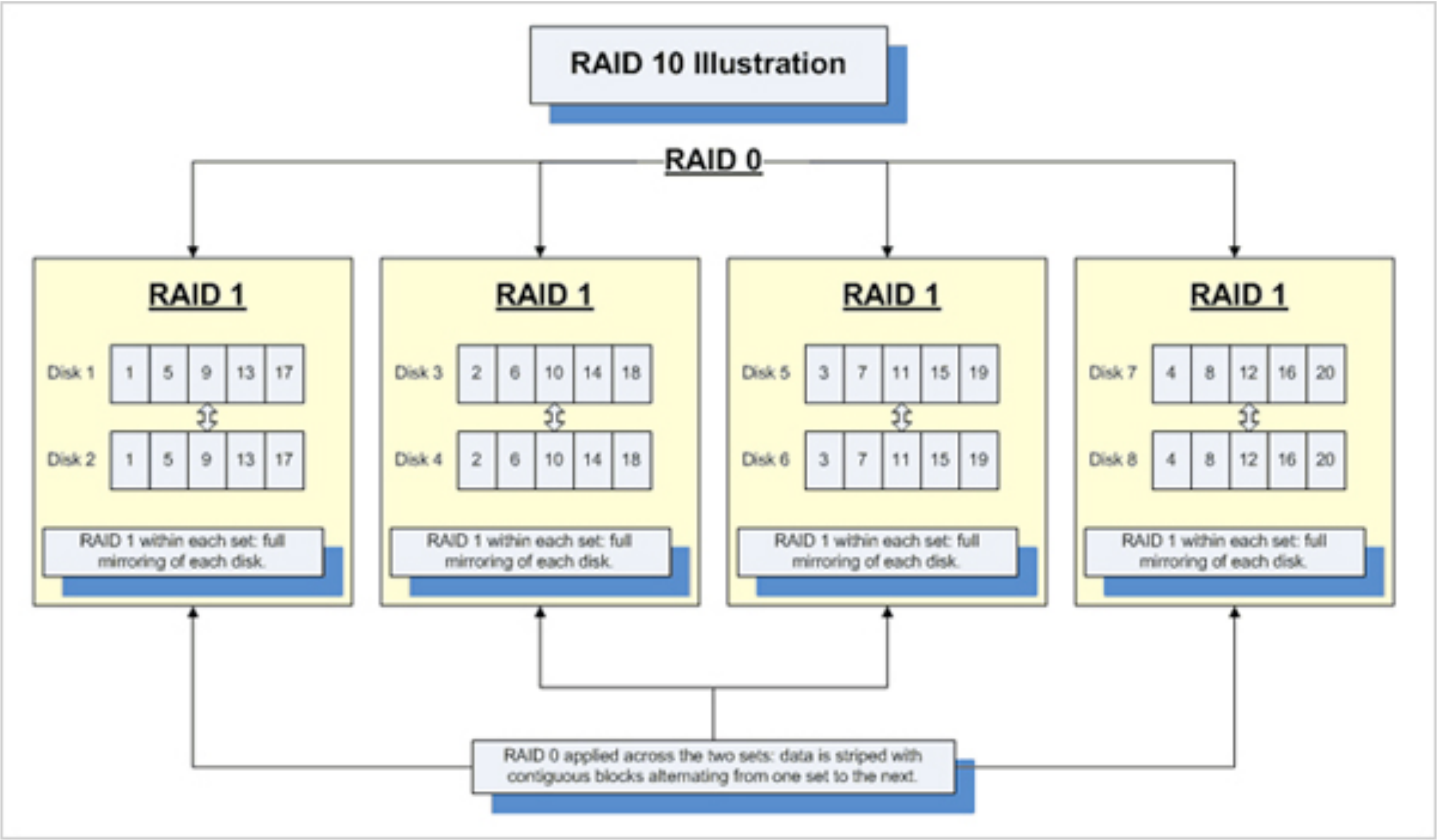
RAID 6 (dual redundancy)



(g) RAID 6 (dual redundancy)

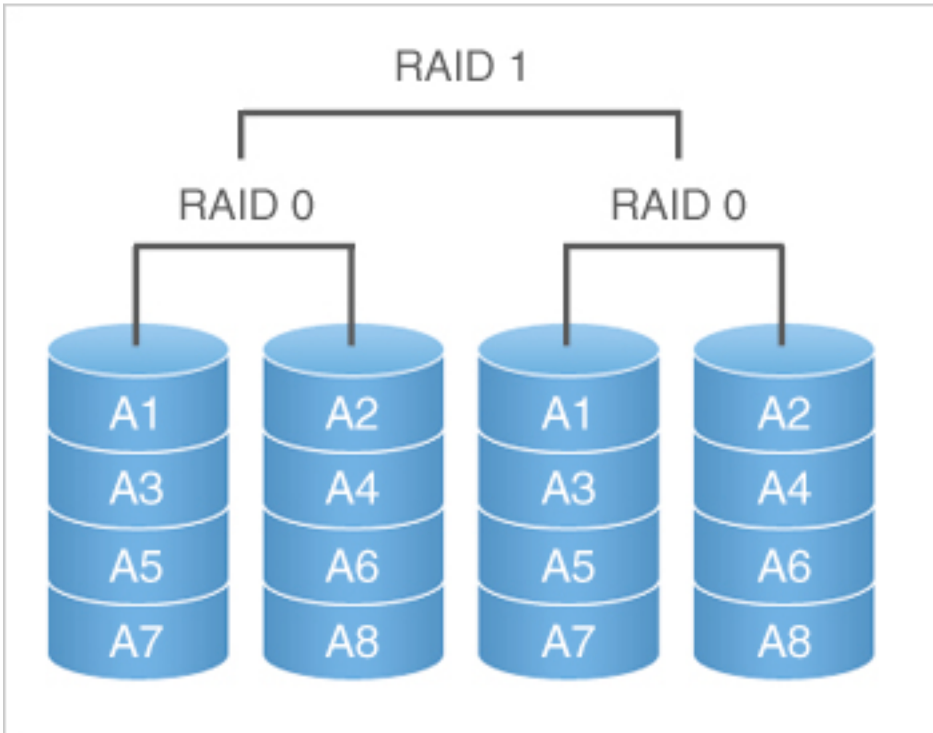
RAID 10

RAID 10 is sometimes also called RAID 1+0



source: http://www.illinoisdataservices.com/raid-10-data-recovery.html

RAID 0+1



source: http://www.illinoisdataservices.com/raid-10-data-recovery.html

Disk Cache

- Buffer in main memory for disk sectors
- Contains a copy of some of the sectors on the disk

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 the reference count is misleading

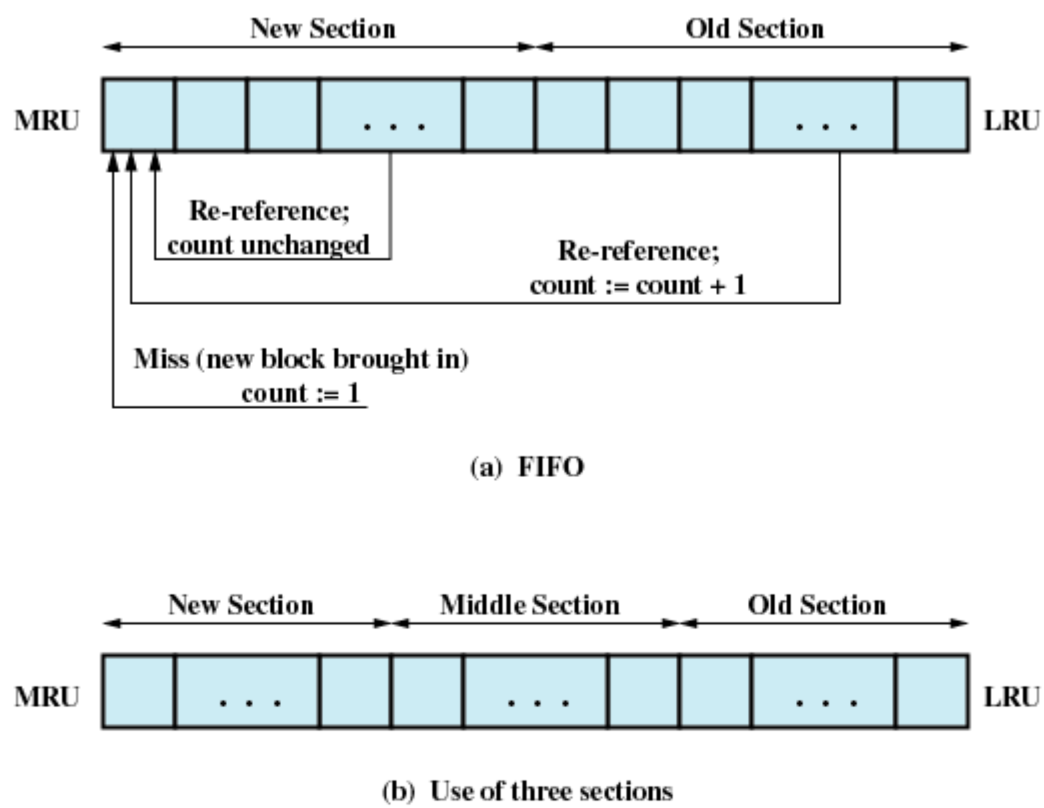


Figure 11.9 Frequency-Based Replacement

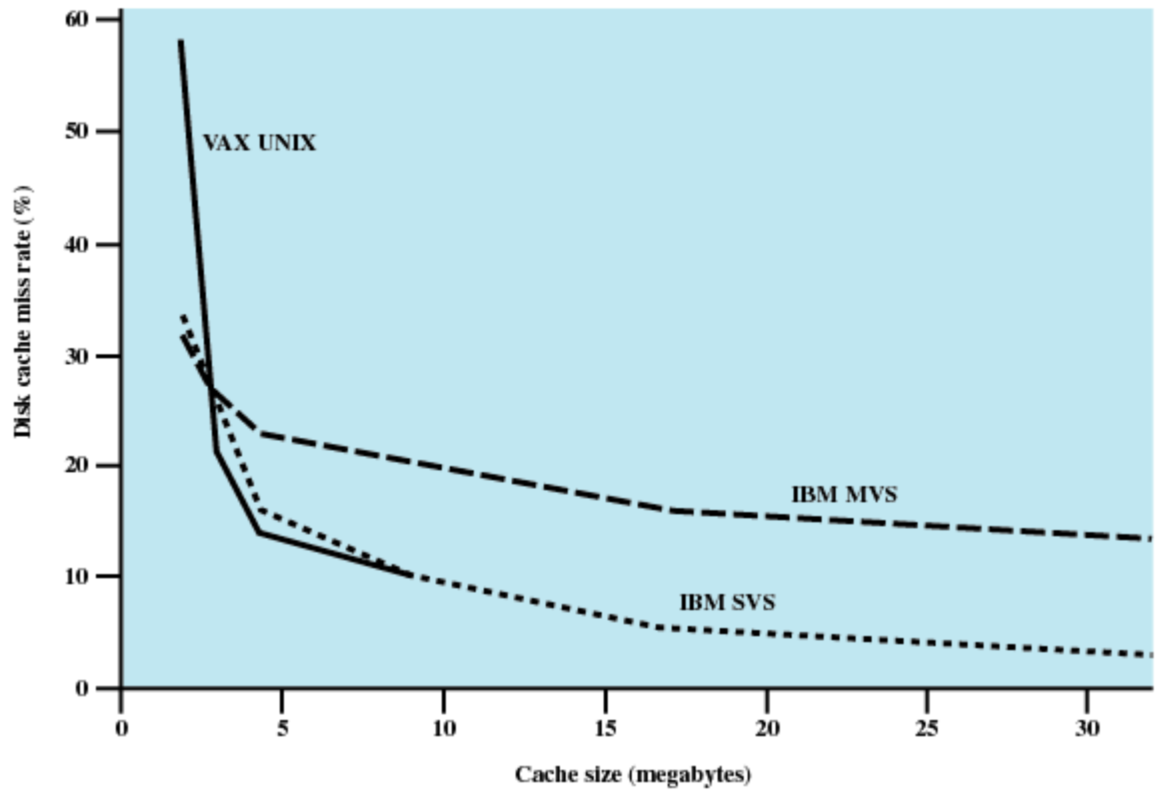


Figure 11.10 Some Disk Cache Performance Results Using LRU

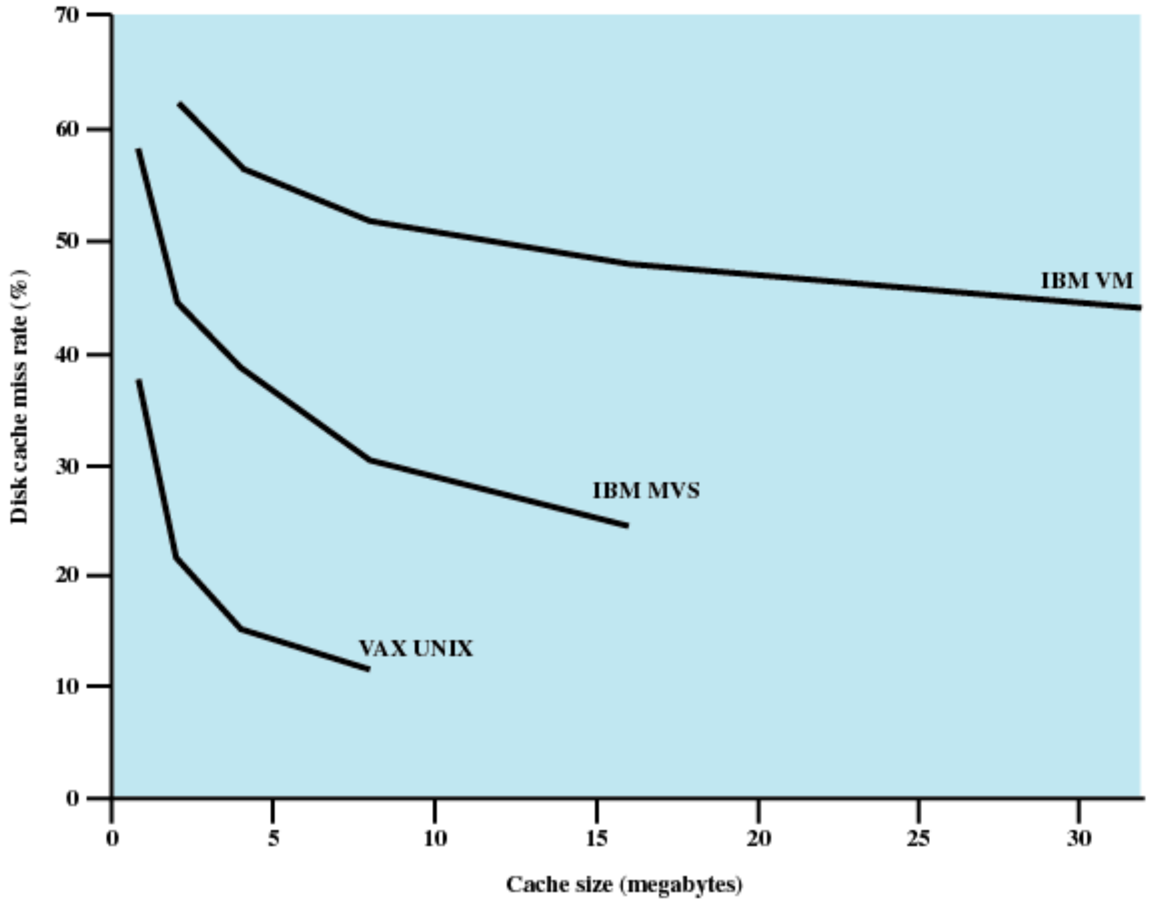


Figure 11.11 Disk Cache Performance Using Frequency-Based Replacement [ROBI90]

UNIX SCR4 I/O

- Each individual device is associated with a special file
- Two types of I/O
  - Buffered
  - Unbuffered

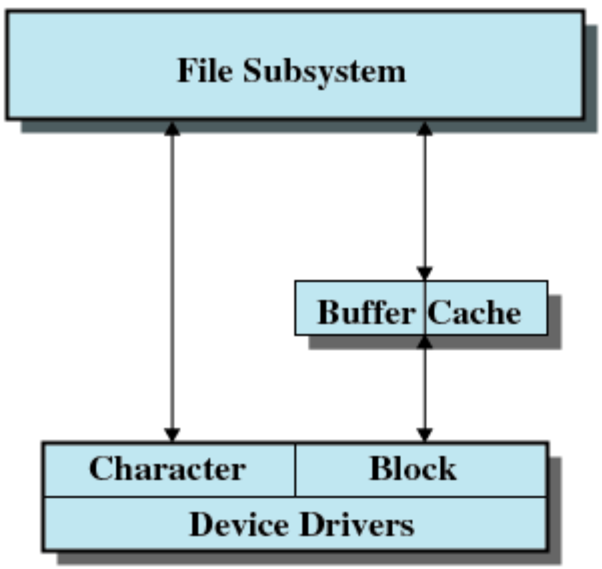


Figure 11.12 UNIX I/O Structure

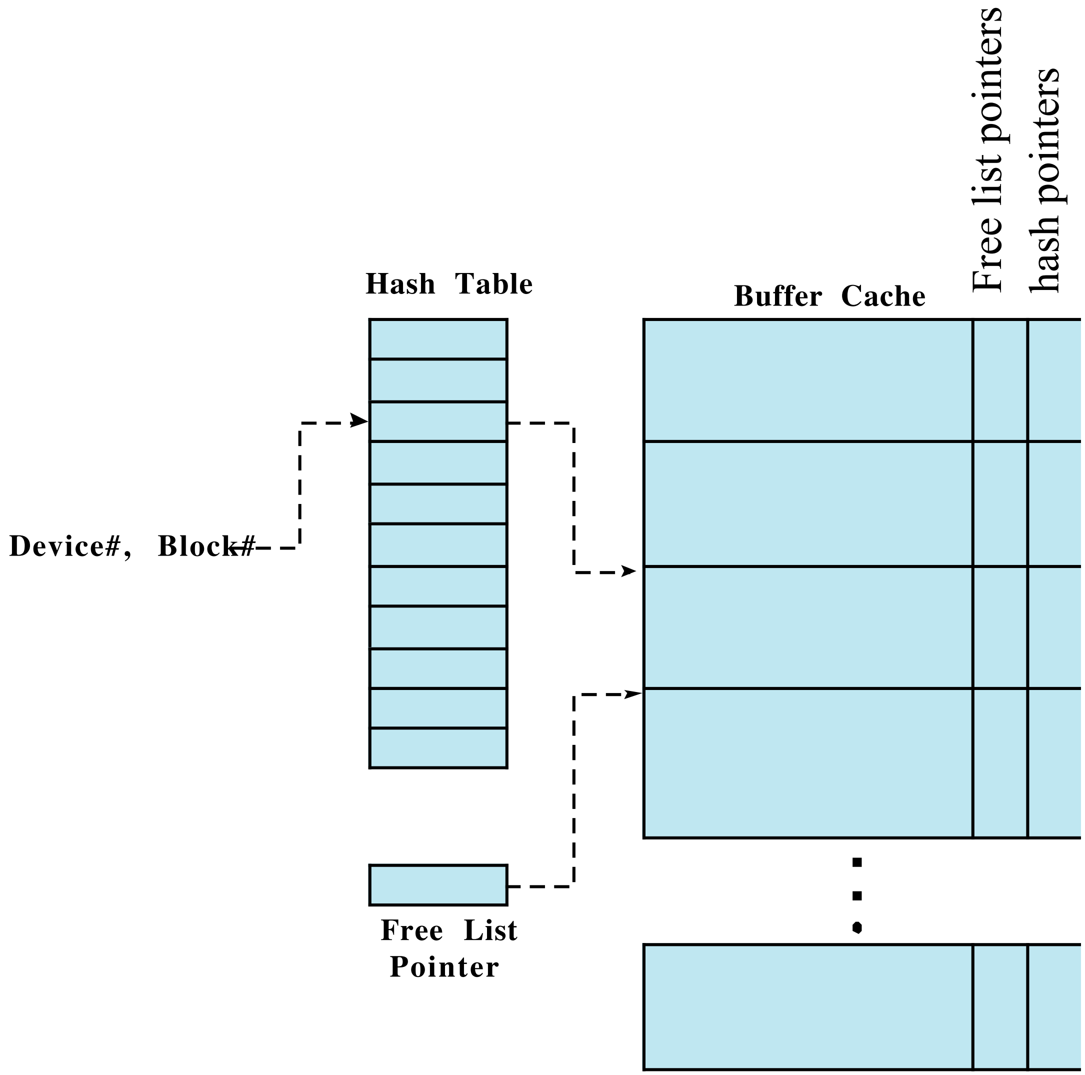


Figure 11.13 UNIX Buffer Cache Organization

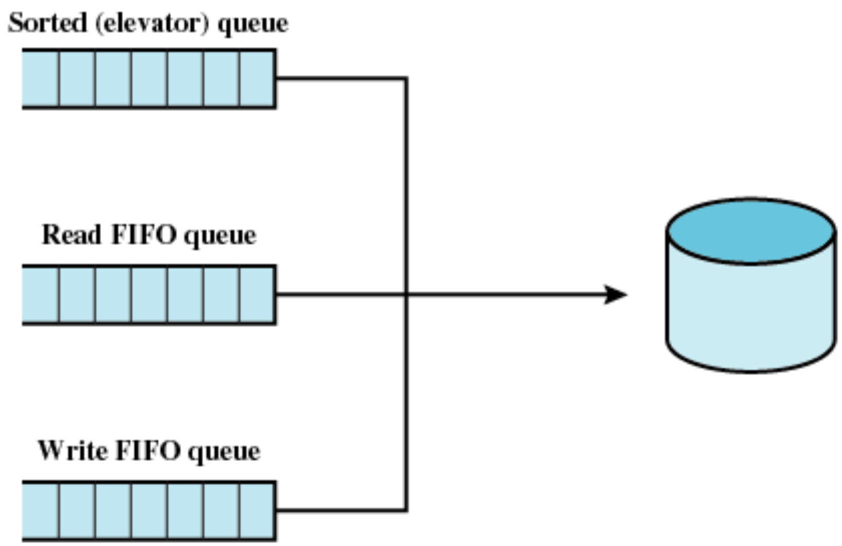
Linux I/O

- Elevator scheduler
  - Maintains a single queue for disk read and write requests
  - Keeps list of requests sorted by block number
  - Drive moves in a single direction to satisfy each request

Linux I/O

- Deadline scheduler
  - Uses three queues
    - Incoming requests
    - Read requests go to the tail of a FIFO queue
    - Write requests go to the tail of a FIFO queue
  - Each request has an expiration time
    - defaults for requests:
      - 0.5s for read
      - 5s for write

Linux I/O



1. Put requests in sorted queue *and* FIFO
   - remove request from both Qs when processed
- Schedule from sorted Q and check expiration date of FIFO entry.
   - if date has expired, schedule from FIFO until "caught up"

Figure 11.14 The Linux Deadline I/O Scheduler

Linux I/O

- Anticipatory I/O scheduler
  - Delay a short period of time after satisfying a read request to see if a new nearby request can be made

Windows I/O

- Basic I/O modules
  - Cache manager
  - File system drivers
  - Network drivers
  - Hardware device drivers

Windows I/O

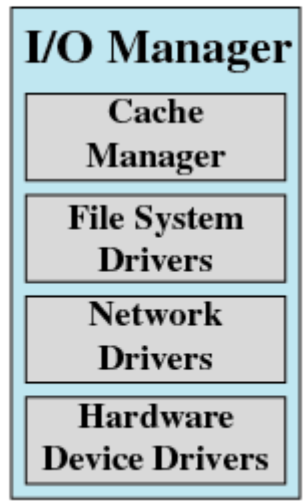


Figure 11.15 Windows I/O Manager