Operating System

Ch11: Mass-storage structure

BeomSeok Kim
Department of Computer Engineering
KyungHee University
passion0822@khu.ac.kr

Mass-Storage

- HDD (Hard Disk Drive)
- SSD (Solid State Disk)
- RAM disk
- Magetic tape

Disk Attachment

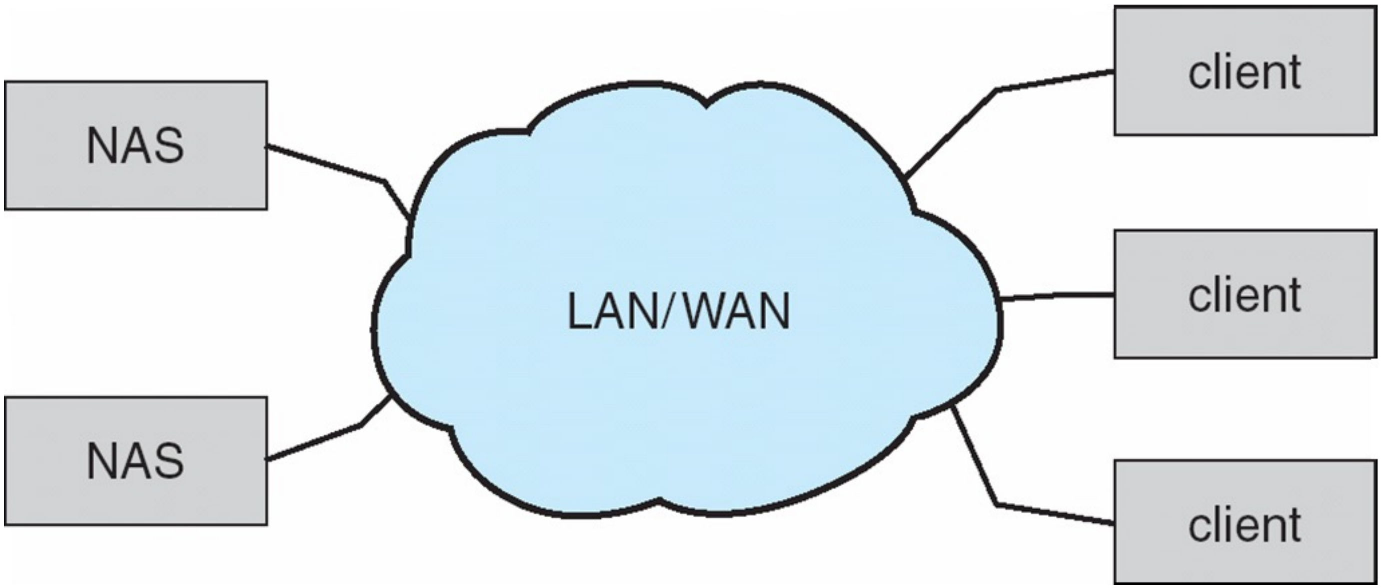
- Disks may be attached one of two ways:

  1. **Host attached** via an I/O port

  2. **Network attached** via a network connection

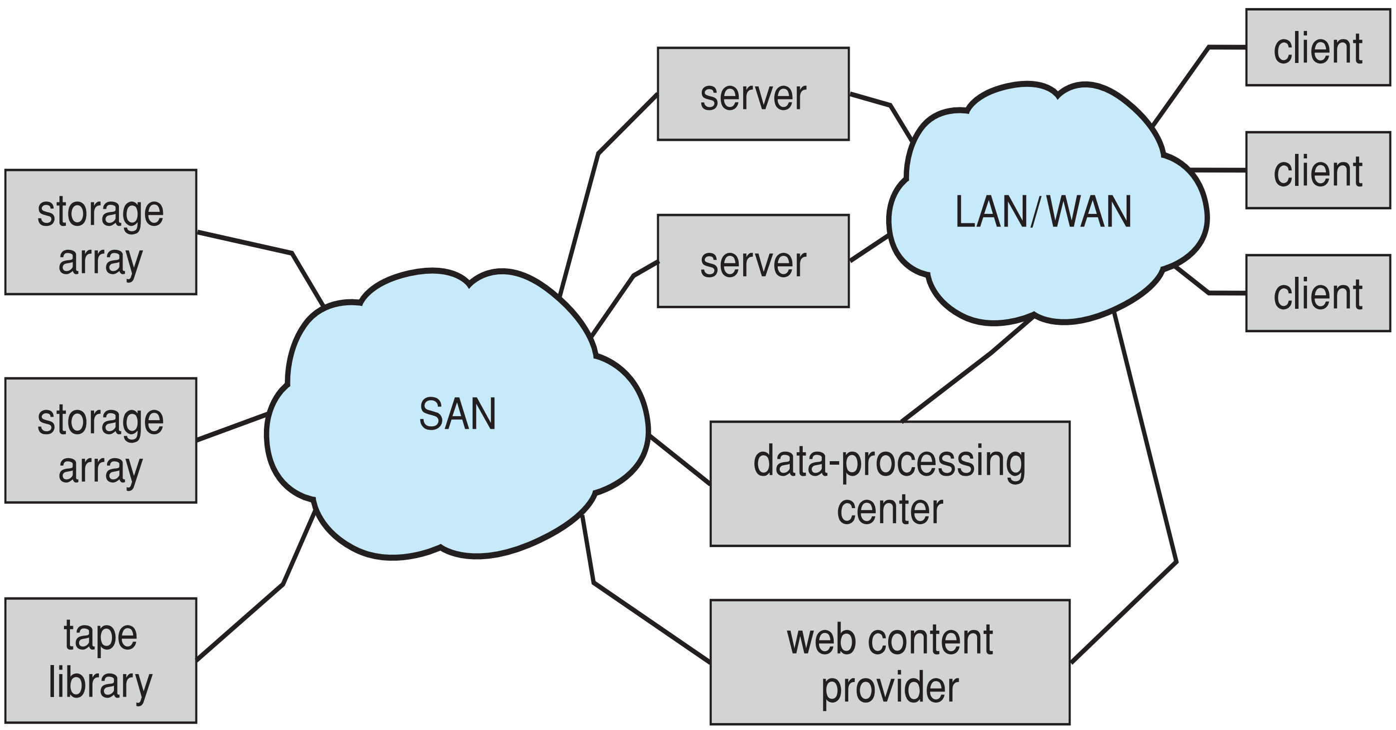
Network-Attached Storage (NAS)

- Accessed via IP networks
- NFS, CIFS, etc.
- File-level access



Storage-Area Network (SAN)

- Accessed via private network dedicated for storages
- Use storage protocols such as SCSI or Fibre channel
- Block-level access
- File systems for SAN is another story (e.g. GFS)



Storage-Area Network (SAN)

- Storage array

Storage Architecture

Protocols

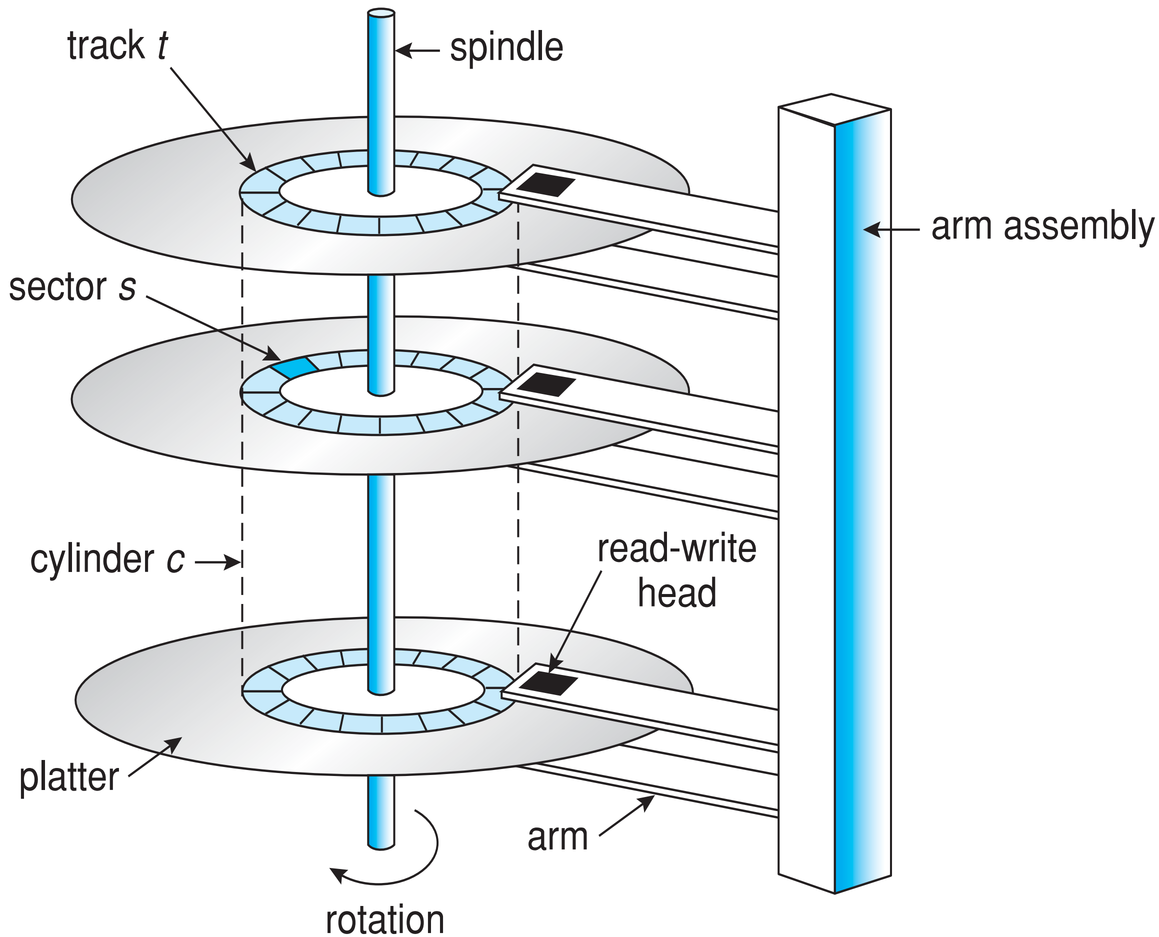
Interconnection

Interconnection	Block	File
Non-IP Networks	IDE / SCSI (Direct) SAN (Storage Area Network: Fibre Channel)	DAFS (Direct Access File System: NFS over VIA)
IP Networks	NBD (Network Block Device) iSCSI (SCSI over TCP/IP)	NAS (Network Attached Storage: NFS, CIFS)

HDD

- Disks and the OS
  - ✓ Disks are messy physical devices:
    - ➢ Errors, bad blocks, missed seeks, etc.
  - ✓ The job of the OS is to hide this mess from higher-level software
    - ➢ Low-level device drivers (initiate a disk read, etc)
    - ➢ Higher-level abstractions (files, databases, etc.)
  - ✓ The OS may provide different levels of disk access to different clients
    - ➢ Physical disk block (surface, cylinder, sector)
    - ➢ Disk logical block (disk block #)
    - ➢ Logical file (filename, block or record or byte #)

HDD Structure



HDD

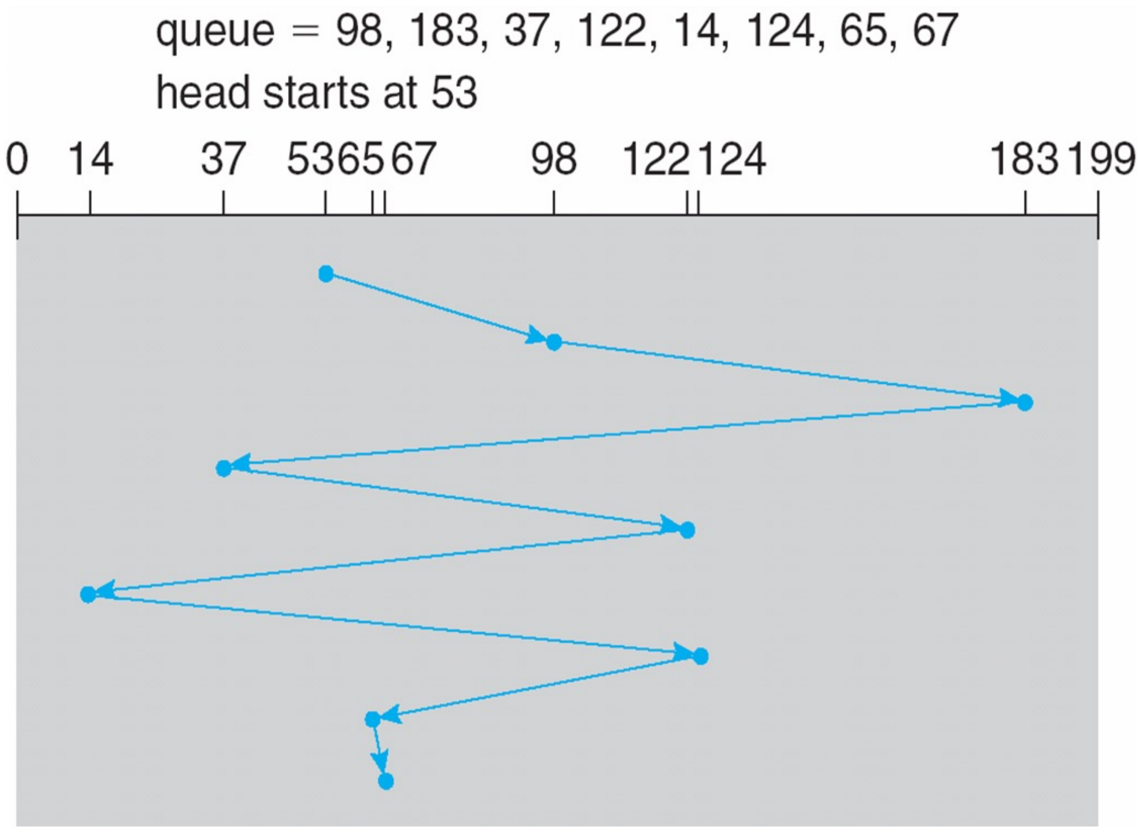
- Interacting with disks
  - ✓ Specifying disk requests requires a lot of info:
    - ➢ Cylinder #, surface #, track #, sector #, transfer size, etc.
  - ✓ Older disks required the OS to specify all of this
    - ➢ The OS needs to know all disk parameters
  - ✓ Modern disks are more complicated
    - ➢ Not all sectors are the same size, sectors are remapped, etc.
  - ✓ Current disks provide a higher-level interface (e.g. SCSI)
    - ➢ The disks exports its data as a logical array of blocks [0..N]
    - ➢ Disk maps logical blocks to cylinder/surface/track/sector
    - ➢ Only need to specify the logical block # to read/write
    - ➢ As a result, physical parameters are hidden from OS

HDD

- Disk performance
  - ✓ Performance depends on a number of steps
    - ➢ Seek: moving the disk arm to the correct cylinder
      $\rightarrow$ depends on how fast disk arm can move (increasing very slowly)
    - ➢ Rotation: waiting for the sector to rotate under head
      $\rightarrow$ depends on rotation rate of disk (increasing, but slowly)
    - ➢ Transfer: transferring data from surface into disk controller, sending it back to the host
      $\rightarrow$ depends on density of bytes on disk (increasing, and very quickly)
  - ✓ Disk scheduling:
    - ➢ Because seeks are so expensive, the OS tries to schedule disk requests that are queued waiting for the disk
    - ➢ Minimize seek time
    - ➢ Seek time $\approx$ seek distance

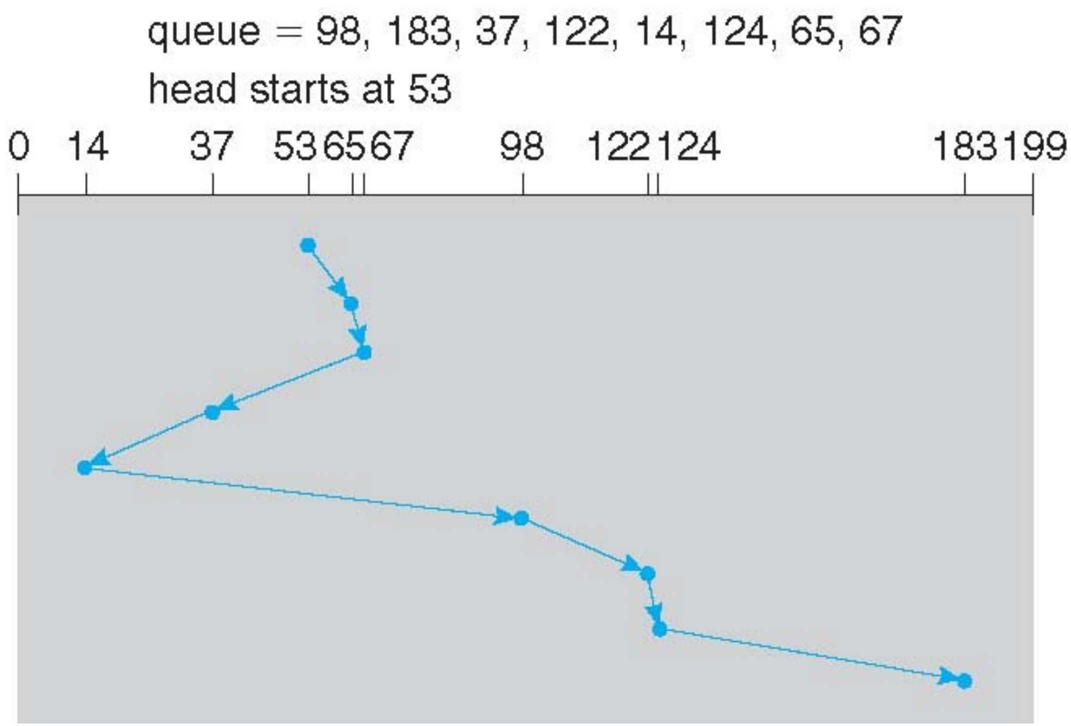
FCFS

- Illustration shows total head movement of 640 cylinders



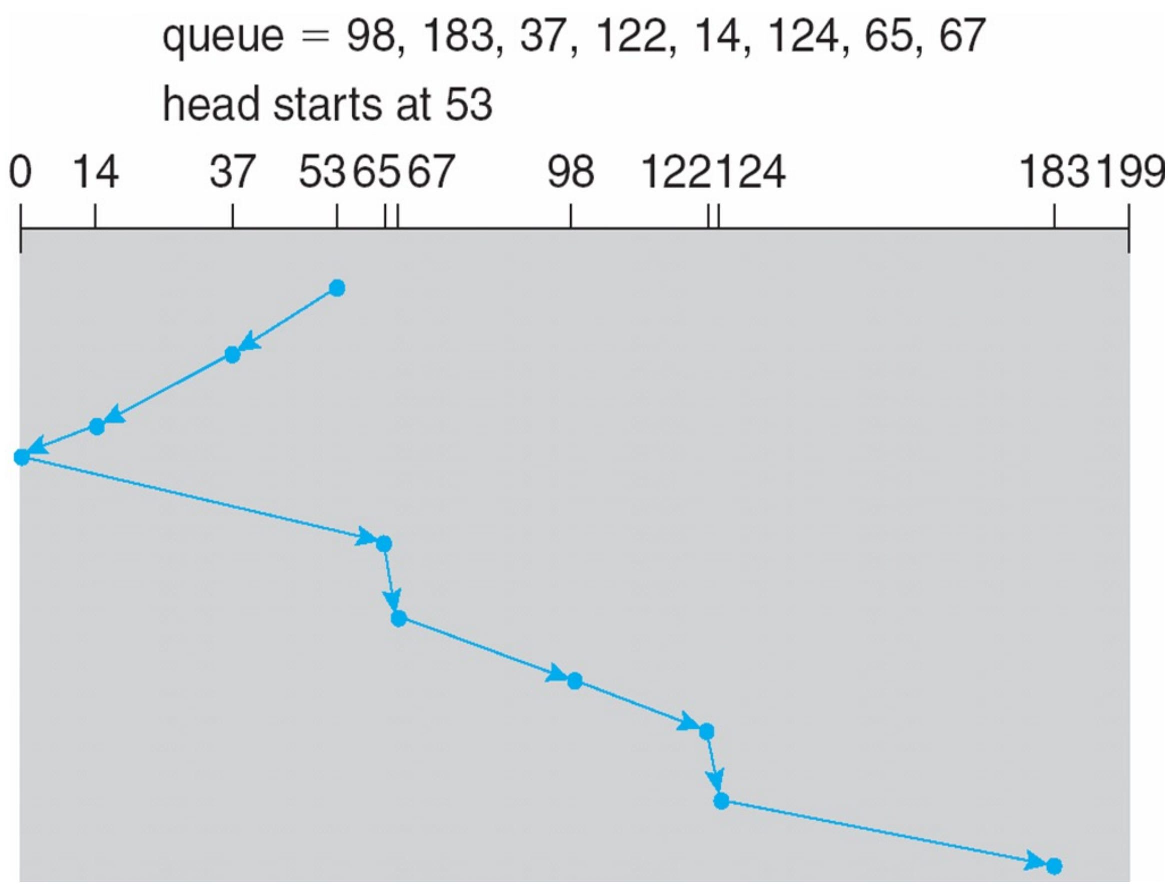
SSTF

- Shortest Seek Time First
  - ✓ selects the request with the minimum seek time from the current head position
  - ✓ a form of SJF scheduling
  - ✓ may cause starvation of some requests



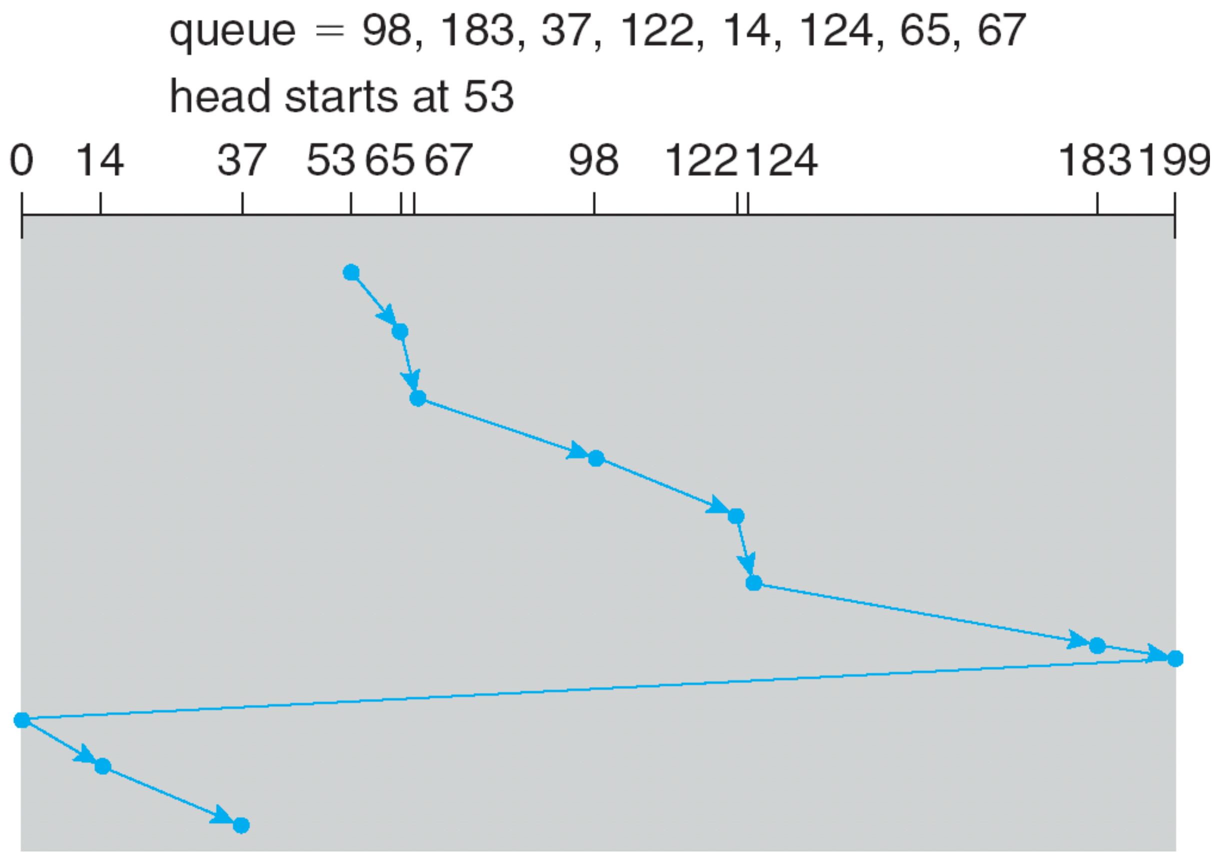
SCAN

- *elevator algorithm*



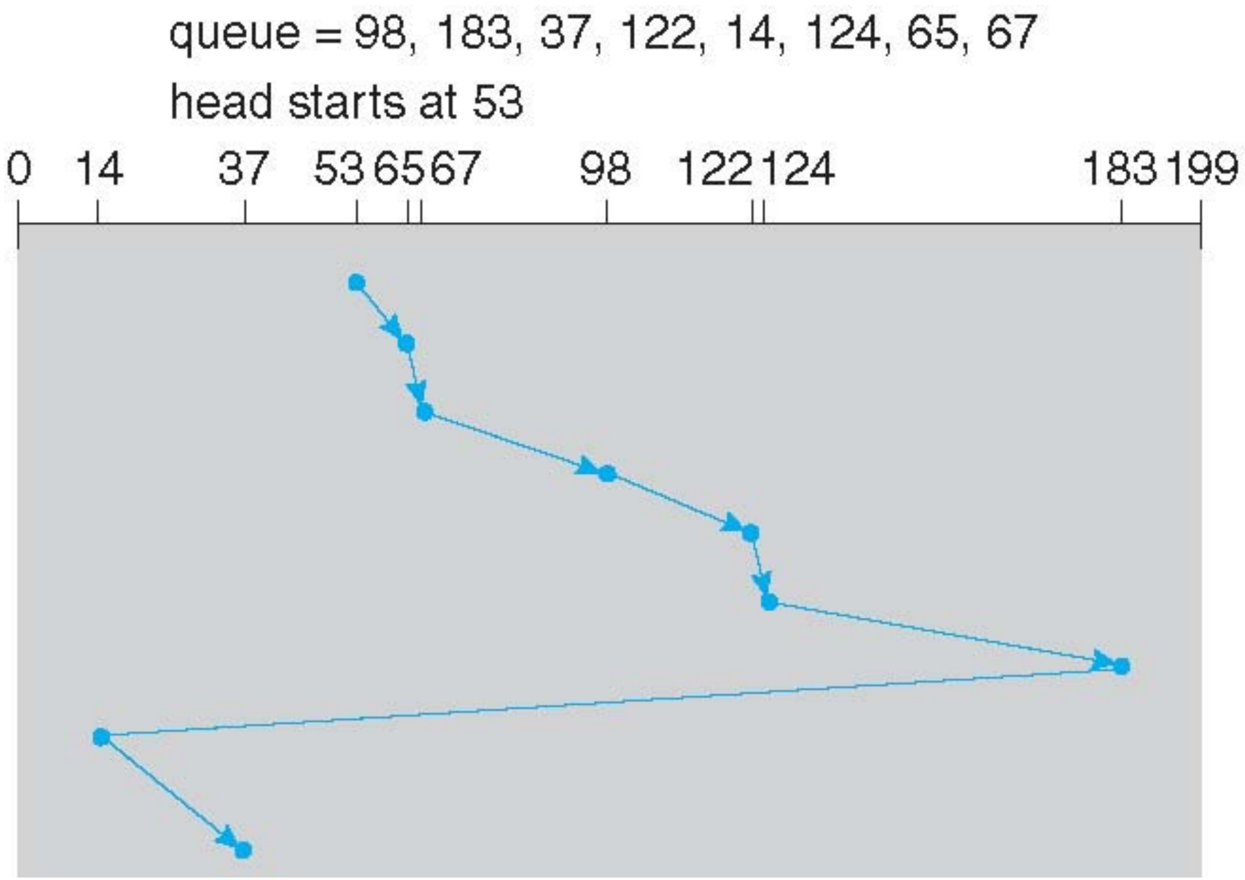
C-SCAN

- Provides a more uniform wait time than SCAN



C-LOOK

- LOOK a version of SCAN, C-LOOK a version of C-SCAN



Selecting a Disk-Scheduling Algorithm

- Performance depends on the number and types of requests
  - ✓ SSTF is common and has a natural appeal
  - ✓ SCAN and C-SCAN perform better for systems that place a heavy load on the disk
- In general, unless there are request queues, disk scheduling does not have much impact
  - ✓ Important for servers, less so for PCs
- Modern disks often do the disk scheduling themselves
  - ✓ Disks know their layout better than OS, can optimize better
  - ✓ Ignores, undoes any scheduling done by OS

Disk Controllers

- Intelligent controllers
  - ✓ Nowadays, most disk controllers are built around a small CPU and have many kilobytes of memory
  - ✓ They run a program written by the controller manufacturer to process I/O requests from the CPU and satisfy them
  - ✓ Intelligent features:
    - ➢ Read-ahead: the current track
    - ➢ Caching: frequently-used blocks
    - ➢ Request reordering: for seek and/or rotational optimality
    - ➢ Request retry on hardware failure
    - ➢ Bad block identification
    - ➢ Bad block remapping: onto spare blocks and/or tracks

Swap-Space Management

- Swap-space
  - ✓ Virtual memory uses disk space as an extension of main memory

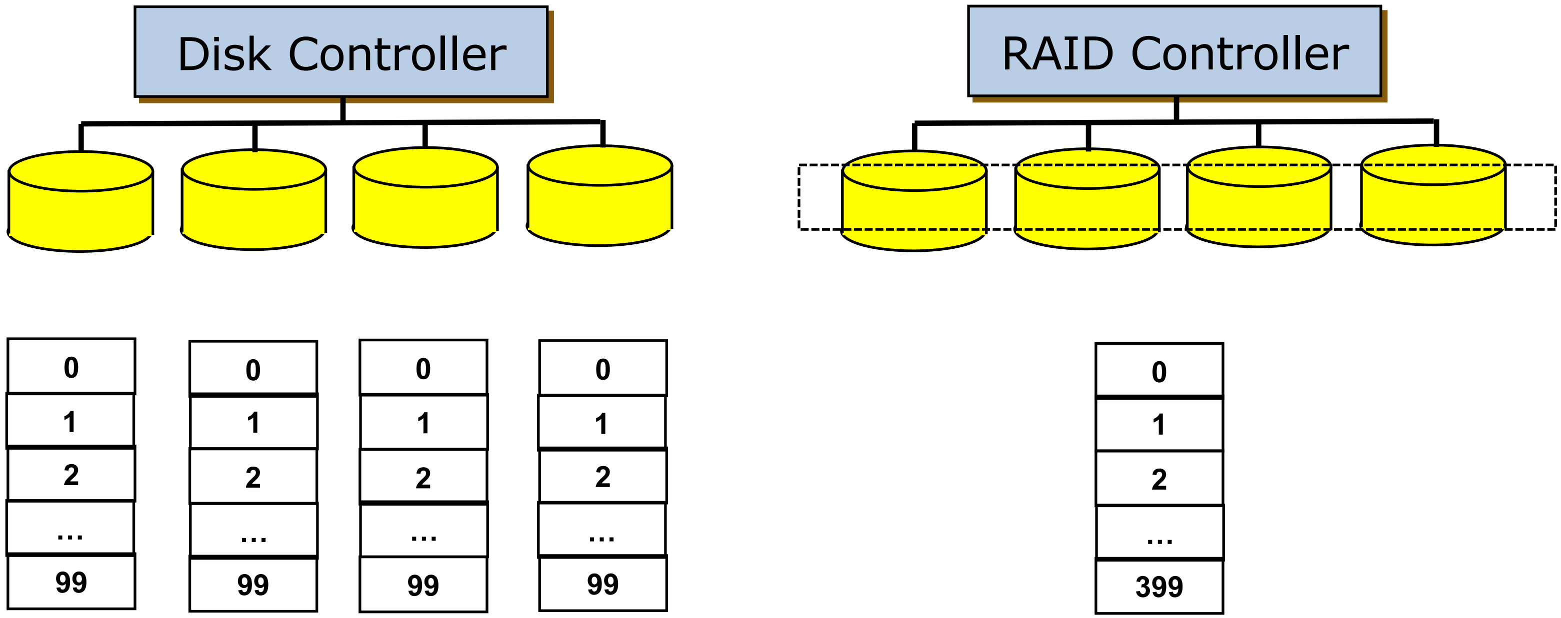
- Swap-space
  - ✓ can be carved out of the normal file system (Windows), or,
  - ✓ more commonly, it can be in a separate disk partition (Linux)

- Swap-space management
  - ✓ 4.3BSD 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

- Redundant Array of Inexpensive Disks
  - ✓ A storage system, not a file system
- Motivations
  - ✓ Use small cheap disks as a cost-effective alternative to large, expensive disks (I = Inexpensive)
  - ✓ Provide higher reliability and higher data-transfer (I = Independent)
- Improving reliability via redundancy
  - ✓ Mirroring (shadowing)
  - ✓ Parity or error-correcting codes
- Improving performance via parallelism
  - ✓ Data striping: bit-level vs. block-level

Multiple Disks vs. RAID



RAID Levels

- RAID 0
  - ✓ Non-redundant striping (no reliability)
  - ✓ Data is broken down into blocks and each block is striped across multiple disks
  - ✓ I/O performance is greatly improved by spreading the I/O load across many channels and drives
  - ✓ Typically used in data rate intensive applications (e.g. video editing)

0	1	2	3
A0	A1	A2	A3
A4	A5	A6	A7
A8	A9	A10	A11
…	…	…	…

RAID Levels

- RAID 1
  - ✓ Mirrored disks
  - ✓ Twice the read transaction rate of single disks, same write transaction rate as single disks
  - ✓ Expensive, highest disk overhead
  - ✓ No rebuild is necessary in case of a disk failure

0	1	2	3	4	5	6	7
A0	B0	C0	D0	A0	B0	C0	D0
A1	B1	C1	D1	A1	B1	C1	D1
A2	B2	C2	D2	A2	B2	C2	D2
...	...	...	...	...	...	...	...

RAID Levels

- RAID 2
  - ✓ Memory-style error-correcting codes (ECC)
  - ✓ Each data word has its Hamming Code ECC word recorded on the ECC disks
  - ✓ On read, the ECC code verifies correct data or corrects single bit errors
  - ✓ ECC is embedded in almost all modern disk drives (e.g. SCSI)

0	1	2	3	4	5	6
A0	A0	A0	A0	Px	Py	Pz
A1	A1	A1	A1	Px	Py	Pz
A2	A2	A2	A2	Px	Py	Pz
A3	A3	A3	A3	Px	Py	Pz
…	…	…	…	…	…	…

RAID Levels

- RAID 3
  - ✓ Bit-interleaved parity
  - ✓ Stripe parity is generated on writes, recorded on the parity disk and checked on reads
  - ✓ Less storage overhead than RAID 2
  - ✓ Requires hardware support for efficient use

0	1	2	3	4
A0	A0	A0	A0	P
A1	A1	A1	A1	P
A2	A2	A2	A2	P
A3	A3	A3	A3	P
…	…	…	…	…

RAID Levels

- RAID 4
  - ✓ Block-interleaved parity
  - ✓ Parity for same rank blocks is generated on writes, recorded on the parity disk and checked on reads
  - ✓ Very good read performance (same as RAID 0)
  - ✓ Writes, however, require parity data be updated each time
  - ✓ No advantages over RAID 5 and does not support multiple simultaneous write operations

0	1	2	3	4
A0	A1	A2	A3	P
A4	A5	A6	A7	P
A8	A9	A10	A11	P
…	…	…	…	…

RAID Levels

- RAID 5
  - ✓ Block-interleaved distributed parity
  - ✓ Parity for blocks in the same rank is generated on writes, recorded in a distributed location
  - ✓ Can speed up small writes in multiprocessing systems, since the parity disk does not become a bottleneck

0	1	2	3	4
A0	A1	A2	A3	P
A4	A5	A6	P	A7
A8	A9	P	A10	A11
A12	P	A13	A14	A15
...	...	...	...	...

RAID Levels

- RAID 6
  - ✓ P+Q redundancy scheme
  - ✓ RAID 5 + extra redundancy information to guard against multiple disk failure
  - ✓ Use error-correcting codes instead of parity

0	1	2	3	4	5
A0	A1	A2	A3	Px	Py
A4	A5	A6	Px	Py	A7
A8	A9	Px	Py	A10	A11
A12	Px	Py	A13	A14	A15
…	…	…	…	…	…

RAID Levels

- RAID 0+1
  - ✓ A set of disks are striped, and then the stripe is mirrored to another, equivalent stripe
  - ✓ RAID 0 provides performance, while RAID 1 provides the reliability
  - ✓ A single drive failure will cause the whole array to become, in essence, a RAID 0 array

RAID1

RAID0 RAID0

0	1	2	3	4	5	6	7
A0	A1	A2	A3	A0	A1	A2	A3
A4	A5	A6	A7	A4	A5	A6	A7
A8	A9	A10	A11	A8	A9	A10	A11
…	…	…	…	…	…	…	…

RAID Levels

- RAID 10 (or RAID 1+0)
  - ✓ Disks are mirrored in pairs, and then the resulting mirror pairs are striped
  - ✓ Reliable better than RAID 0+1
    - ➢ RAID 0+1 is fault tolerant as long as the second through n-th disk is on the same stripe
    - ➢ RAID 1+0 is fault tolerant as long as no two disks are part of the same mirror

RAID0

R1 R1 R1 R1

0	1	2	3	4	5	6	7
A0	A0	A1	A1	A2	A2	A3	A3
A4	A4	A5	A5	A6	A6	A7	A7
A8	A8	A9	A9	A10	A10	A11	A11
…	…	…	…	…	…	…	…

Thank You!
Q&A