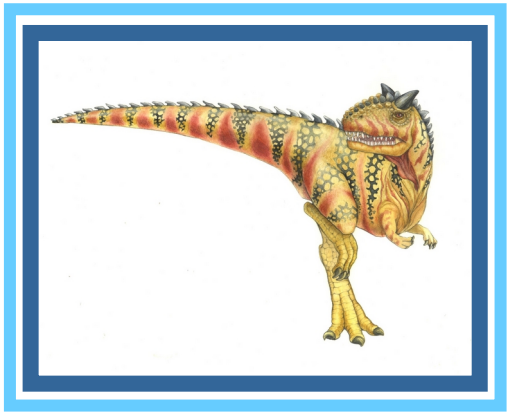
Chapter 11: Mass-Storage Systems



Chapter 11: Mass-Storage Systems

- Overview of Mass Storage Structure
- Storage Attachment
- Secondary Storage I/O Scheduling
- Storage Device Management
- Swap-Space Management
- RAID Structure

Disk Scheduling

- The operating system is responsible for using hardware efficiently — for the disk drives, this means having a fast access time and disk bandwidth
- Minimize seek time
- Seek time ≈ seek distance
- Disk **bandwidth** is the total number of bytes transferred, divided by the total time between the first request for service and the completion of the last transfer

Disk Scheduling (Cont.)

- There are many sources of disk I/O request
  - OS
  - System processes
  - Users processes
- I/O request includes input or output mode, disk address, memory address, number of sectors to transfer
- OS maintains queue of requests, per disk or device
- Idle disk can immediately work on I/O request, busy disk means work must be queued.
  - Optimization algorithms only make sense when a queue exists

Disk Scheduling (Cont.)

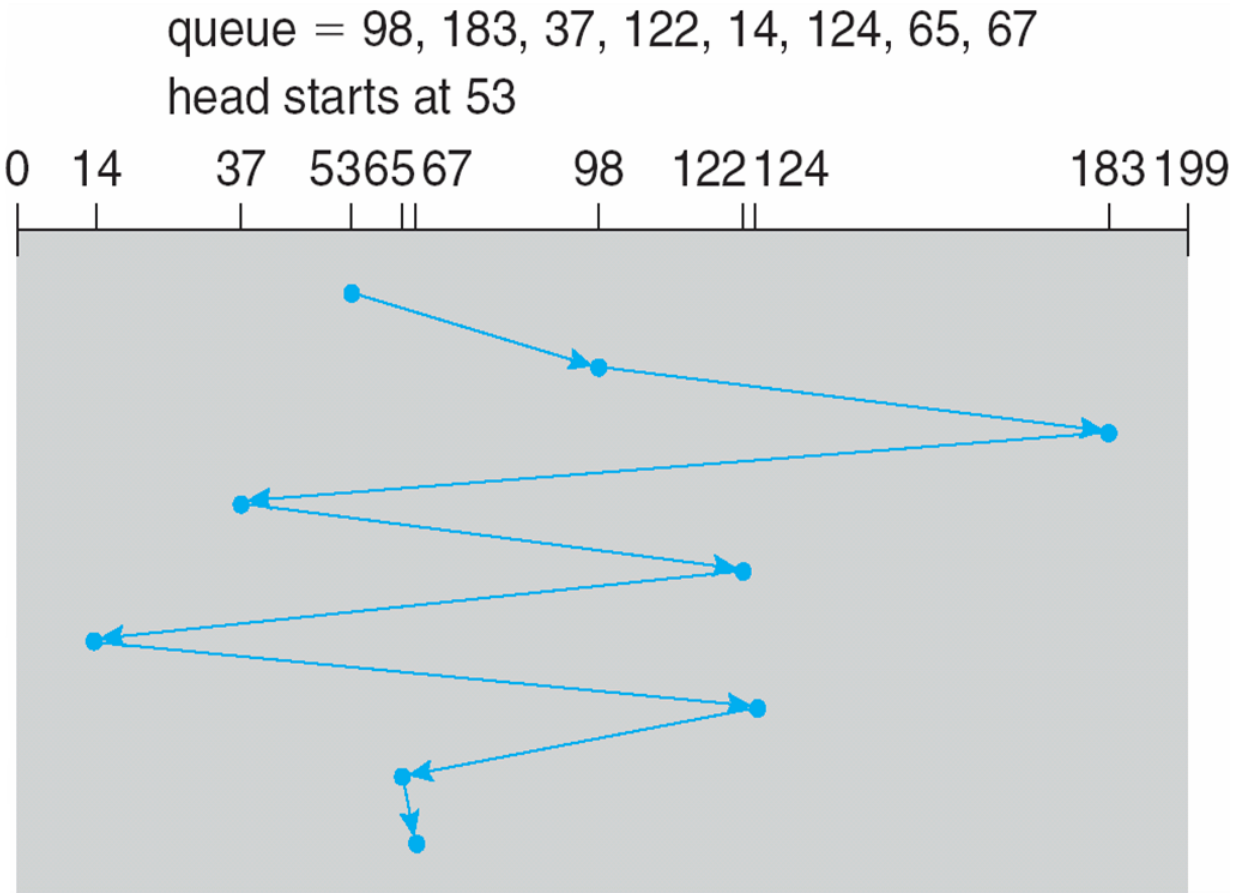
- Note that drive controllers have small buffers and can manage a queue of I/O requests (of varying "depth")
- Several algorithms exist to schedule the servicing of disk I/O requests
- The analysis is true for one or many platters
- We illustrate scheduling algorithms with a request queue (0-199)

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

Head pointer 53

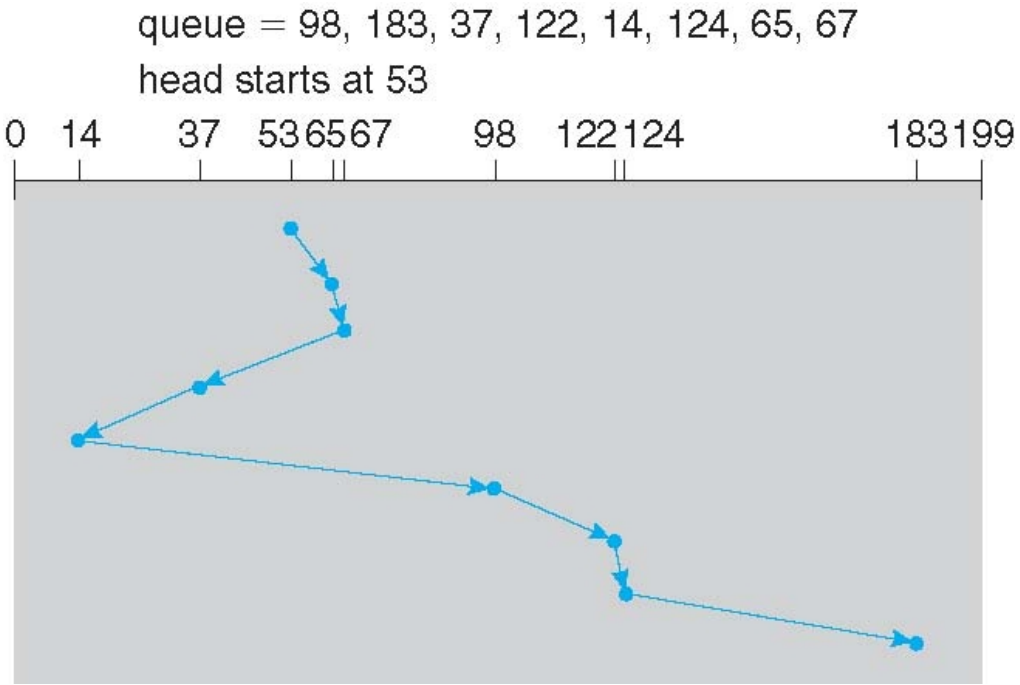
FCFS

Illustration shows total head movement of 640 cylinders



SSTF

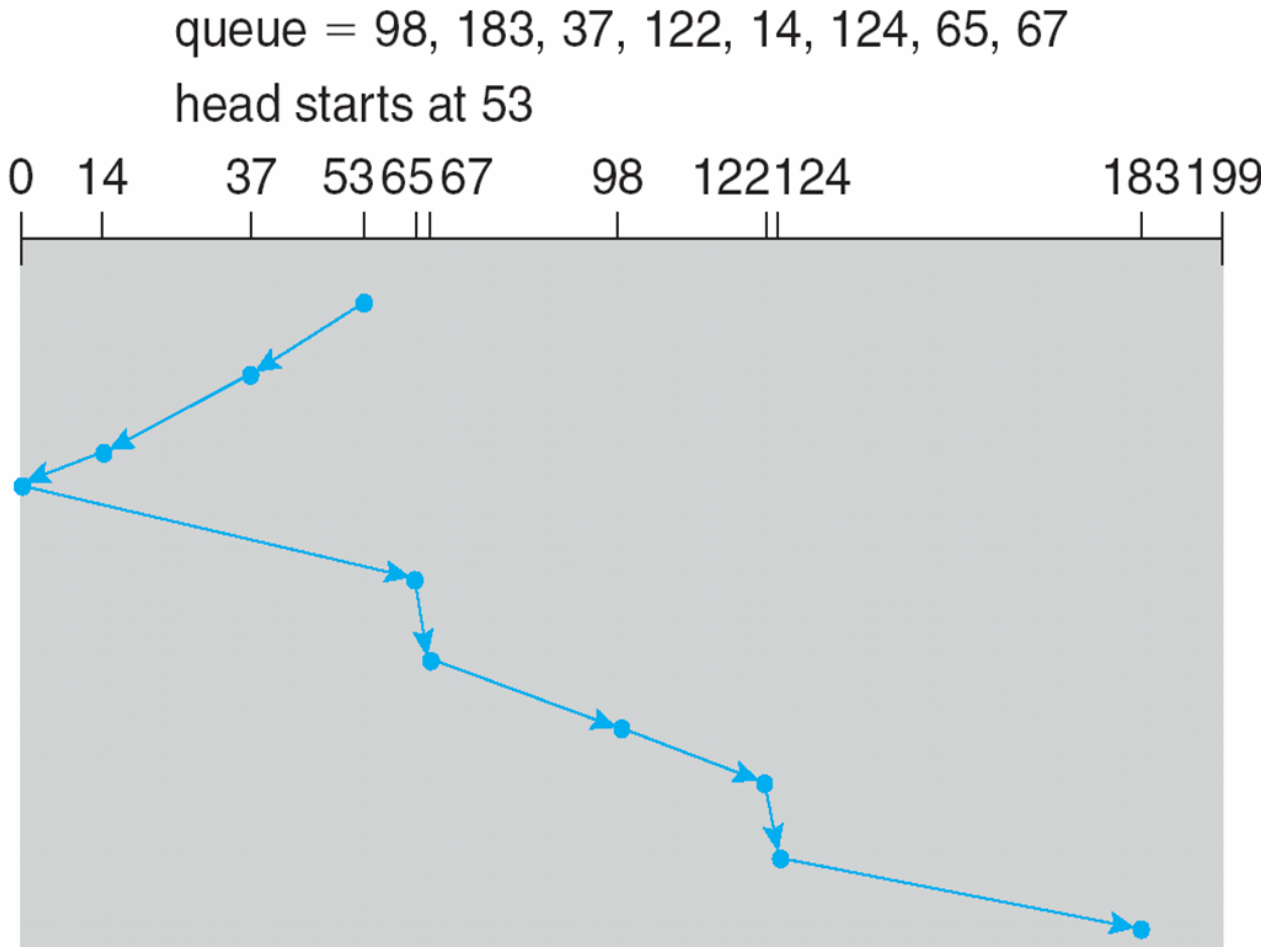
- Shortest Seek Time First -- selects the request with the minimum seek time from the current head position
- SSTF scheduling is a form of SJF scheduling; may cause starvation of some requests
- Illustration shows total head movement of 236 cylinders



SCAN

- **SCAN algorithm**. 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.
- Sometimes it is called the **elevator algorithm**
- Illustration shows total head movement of 208 cylinders
- But note that if requests are uniformly dense, largest density at other end of disk and those wait the longest

SCAN (Cont.)


queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
0
14
37
53
65
67
98
122
124
183
199

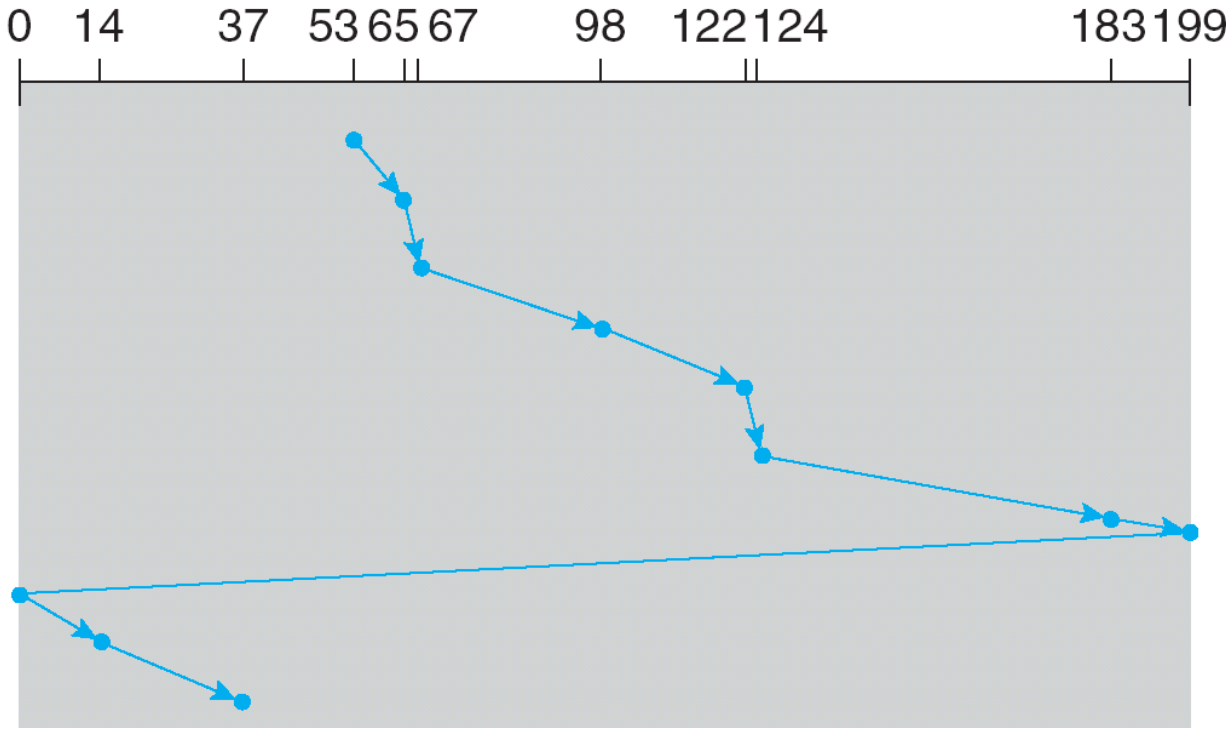
C-SCAN

- Provides a more uniform wait time than SCAN
- 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
- Treats the cylinders as a circular list that wraps around from the last cylinder to the first one

C-SCAN (Cont.)

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

head starts at 53



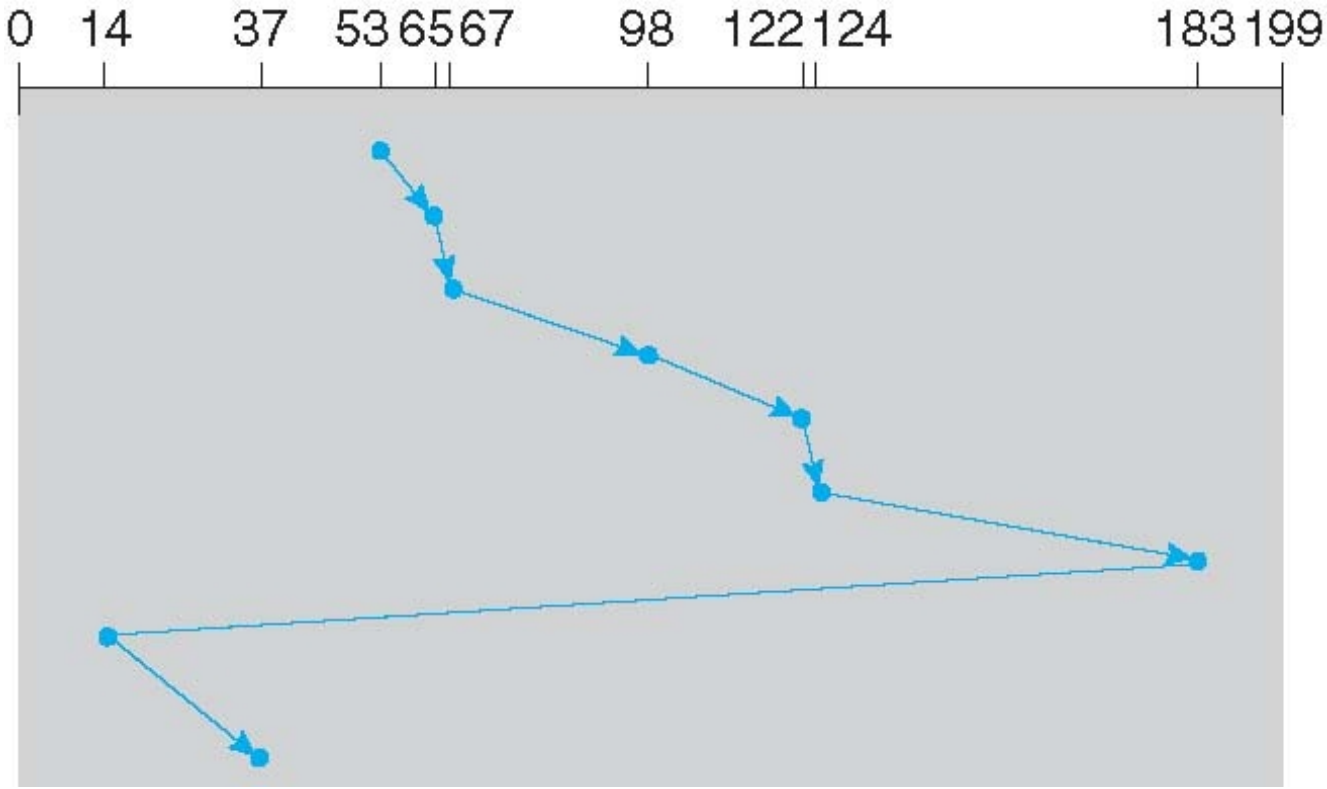
LOOK and C-LOOK

- **LOOK** is a version of 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** a version of C-SCAN. Arm only goes as far as the last request in one direction, then reverses direction immediately, without first going all the way to the end of the disk

C-LOOK (Cont.)

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

head starts at 53



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
  - Less starvation
- Performance depends on the number and types of requests
- Requests for disk service can be influenced by the file-allocation method
  - And metadata layout

Disk-Scheduling Algorithm (Cont.)

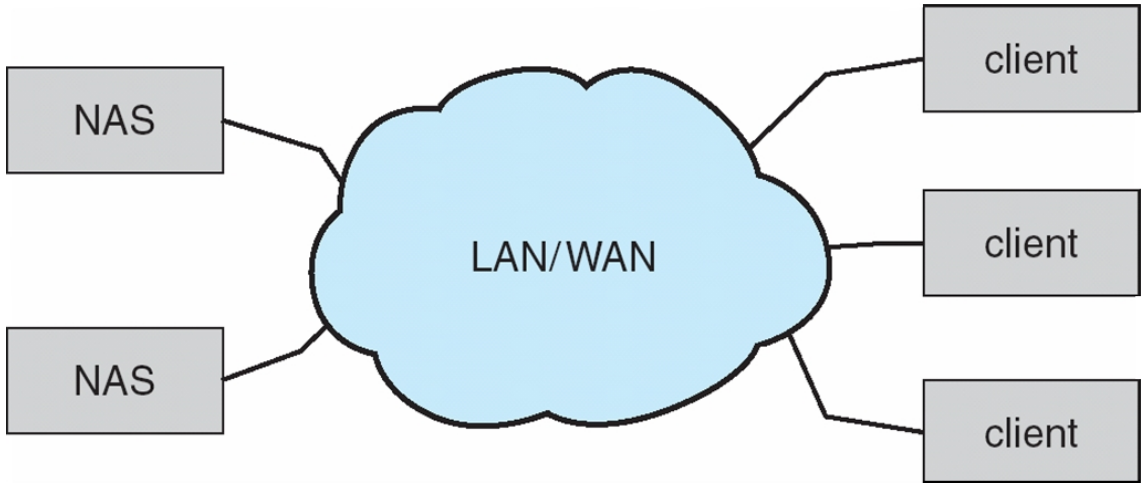
- The disk-scheduling algorithm should be written as a separate module of the operating system, allowing it to be replaced with a different algorithm if necessary
- Either SSTF or LOOK is a reasonable choice for the default algorithm
- What about rotational latency?
  - Difficult for OS to calculate
- How does disk-based queuing effect OS queue ordering efforts?

Network-Attached Storage

- Network-attached storage (**NAS**) is storage made available over a network rather than over a local connection (such as a bus)
  - Remotely attaching to file systems
  - Manages storage, provides data management, has network interfaces, and talks network storage protocols
- NFS and CIFS are common protocols
- Implemented via remote procedure calls (RPCs) between host and storage over typically TCP or UDP on IP network

Network-Attached Storage (Cont.)

- **iSCSI** protocol uses IP network to carry the SCSI protocol
  - Remotely attaching to devices (blocks)

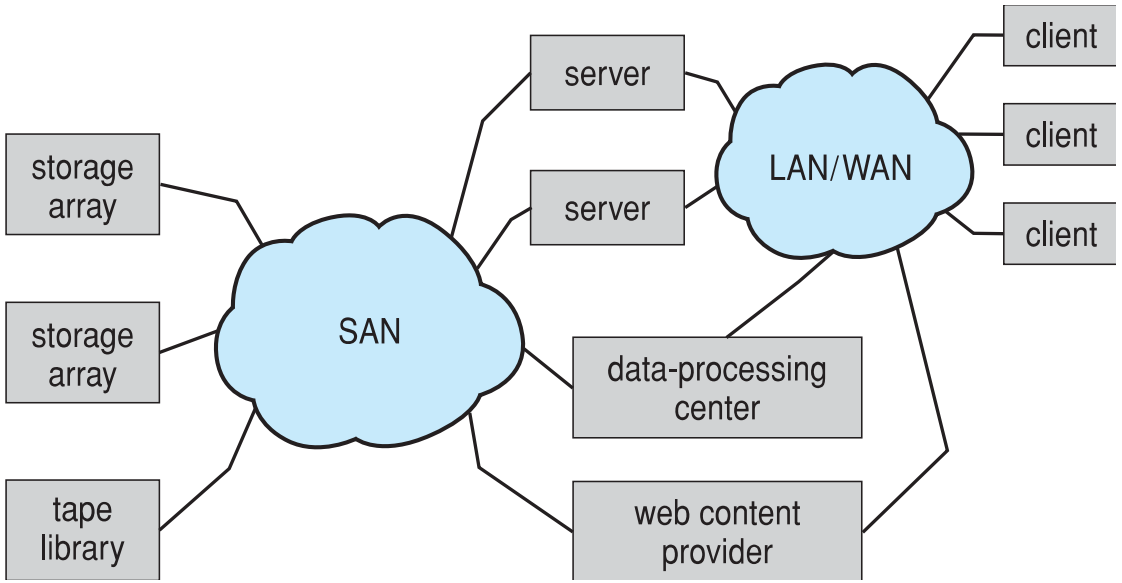


Cloud Storage

- Similar to NAS
  - Provides storage across the network
  - Usually not owned by the company or user, but provided for a fee (based on time, storage capacity used, I/O done, etc.)
  - But across a WAN rather than a LAN
  - Frequently too slow, more prone to connection interruption than NAS so CIFS, NFS, iSCSI possibly but less used
  - Frequently use their own APIs, and apps that use those APIs to do I/O
    - Dropbox, Microsoft OneDrive, Apple iCloud, etc.

Storage Area Network

- Common in large storage environments
- Multiple hosts attached to multiple storage arrays - flexible



Reliability and Redundancy

- **Mean time to failure**. The average time it takes a disk to fail.
- **Mean time to repair.** The time it takes (on average) to replace a failed disk and restore the data on it.
- **Mirroring.** Copy of a disk is duplicated on another disk.
  - Consider disk with 100,000 hours of mean time to failure and 10 hour mean time to repair
    - Mean time to data loss is $100{,}000^2 / (2 * 10) = 500 * 10^6$ hours, or 57,000 years If mirrored disks fail independently,
- Several improvements in disk-use techniques involve the use of multiple disks working cooperatively

RAID Structure

- RAID – redundant array of inexpensive disks
- Use of many disks Increases the **mean time to failure.**
  - 100 disks with MTF of 100,000 hours.
  - 100,000/100 = 1,000 hours or 41.66 days.
- Solution is to have data redundancy over the 100 disks.
- **Disk striping.** Splitting the bits (or blocks) across multiple disks
  - **bit-level striping**. The bits of a byte are split across multiple disks.
  - **block-level striping**. The blocks of a file byte are split across multiple disks.
- For example, if we have an array of eight disks, we write bit "*i*" of each byte to disk "*i*". The array of eight disks can be treated as a single disk with sectors that are eight times the normal size and, more important, that have eight times the access rate.

RAID Structure (Cont.)

- RAID schemes improve performance and improve the reliability of the storage system by storing redundant data
- RAID within a storage array can still fail if the array fails:
  - Replication of the data between arrays is common -- automatic duplication of writes between separate sites
    - For redundancy and disaster recovery
    - Can be synchronous or asynchronous
  - Frequently, a small number of hot-spare disks are left unallocated, automatically replacing a failed disk and having data rebuilt onto them
  - Hot spare disk is not used for storing data. It is configured to be used as a replacement in case of a disk failure.
- Snapshot is a view of file system before a set of changes take place (i.e., at a point in time). More in Ch 12

RAID Levels

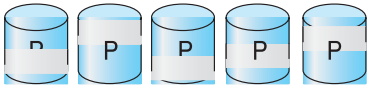
(a) RAID 0: non-redundant striping.

(b) RAID 1: mirrored disks.

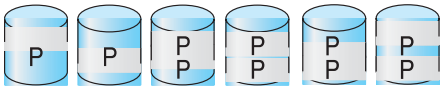
(c) RAID 2: memory-style error-correcting codes.

(d) RAID 3: bit-interleaved parity.

(e) RAID 4: block-interleaved parity.



(f) RAID 5: block-interleaved distributed parity.



(g) RAID 6: P + Q redundancy.