Secondary Storage Devices

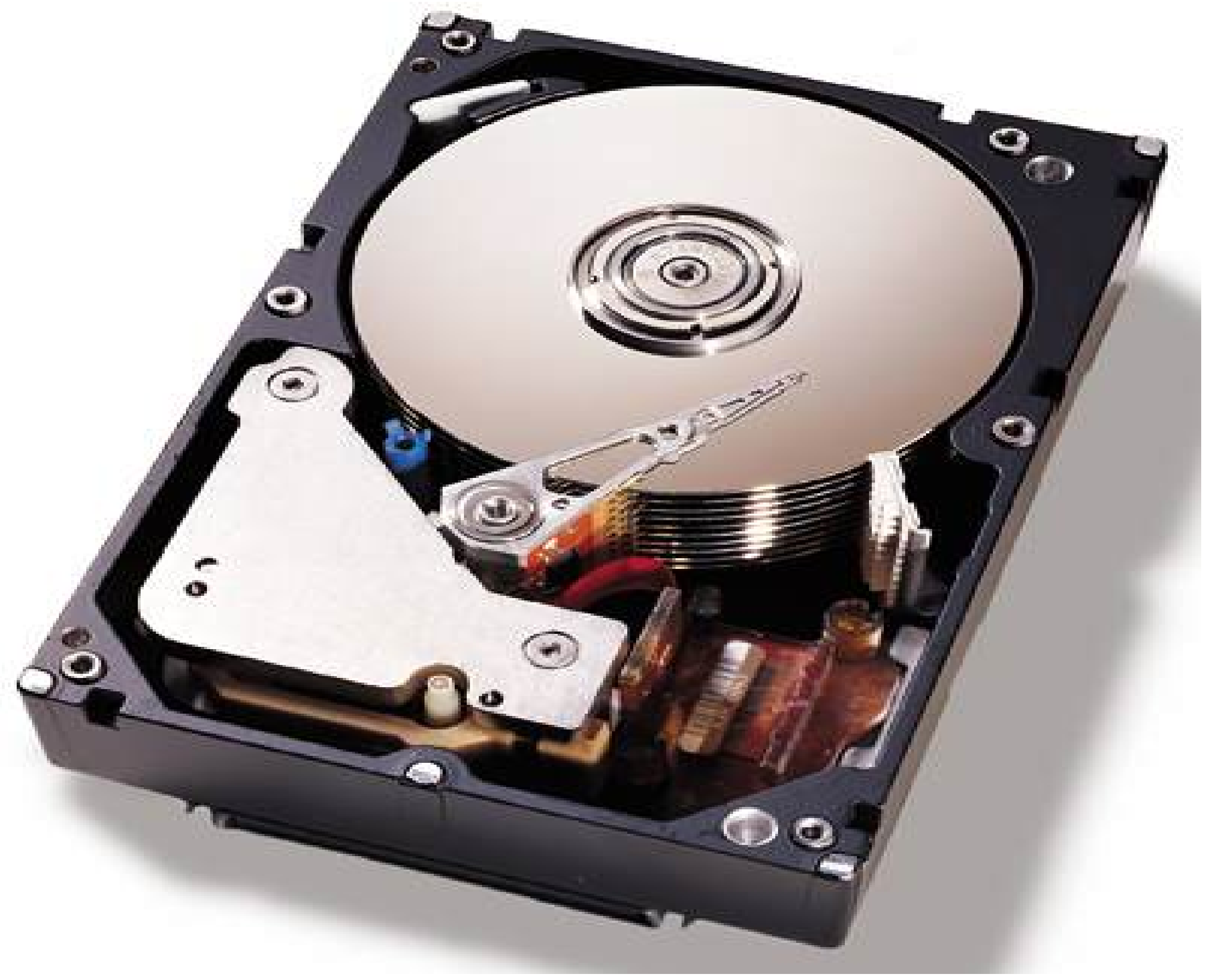
Secondary Storage Devices

- Two major types of storage devices:

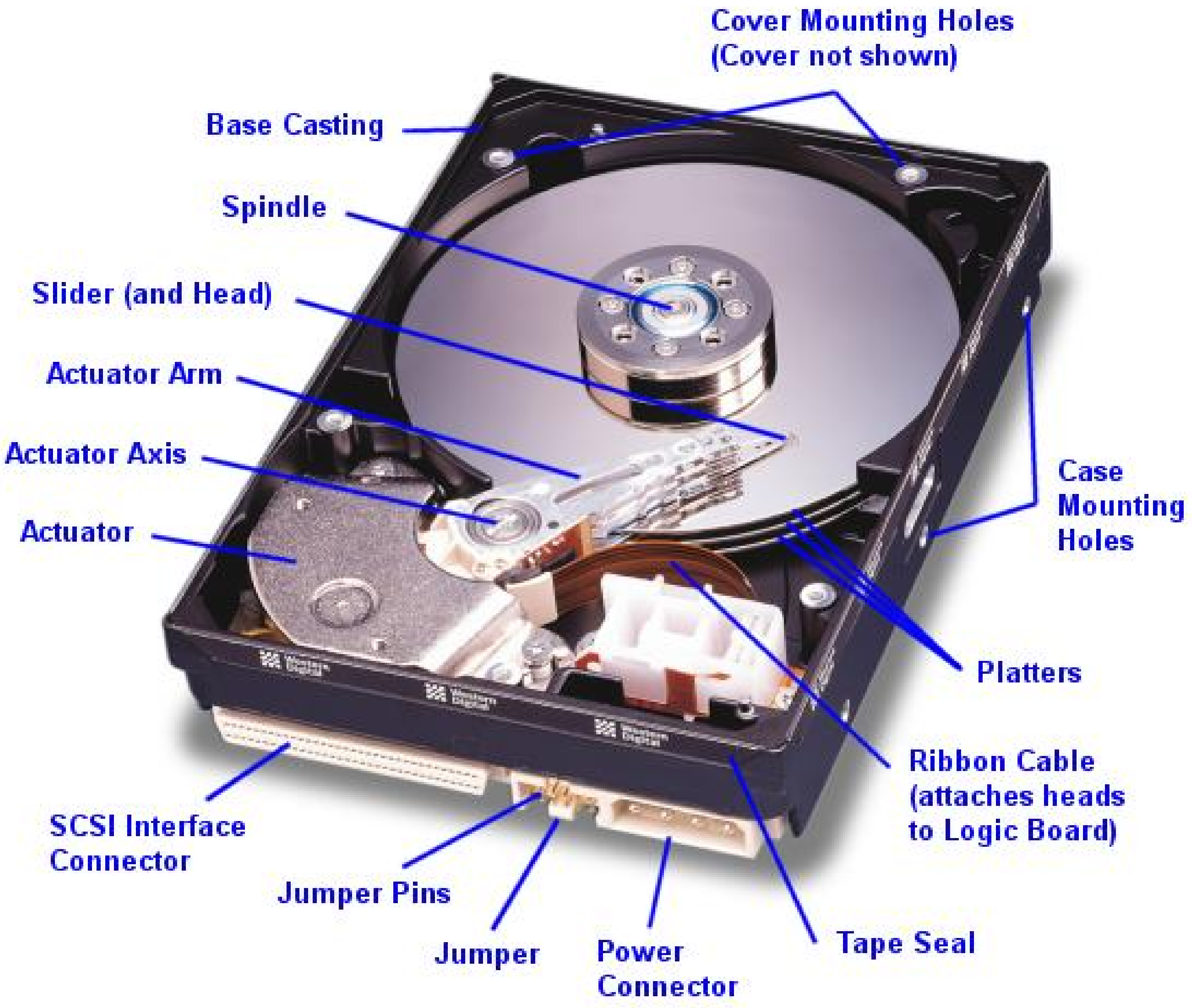
2. Direct Access Storage Devices (DASDs)
   - Magnetic Disks
     Hard disks (high capacity, low cost per bit)
     Floppy disks (low capacity, slow, cheap)
   - Optical Disks
     CD-ROM = (Compact disc, read-only memory)
     DVD
3. Serial Devices
   - Magnetic tapes (very fast sequential access)

Magnetic Disks

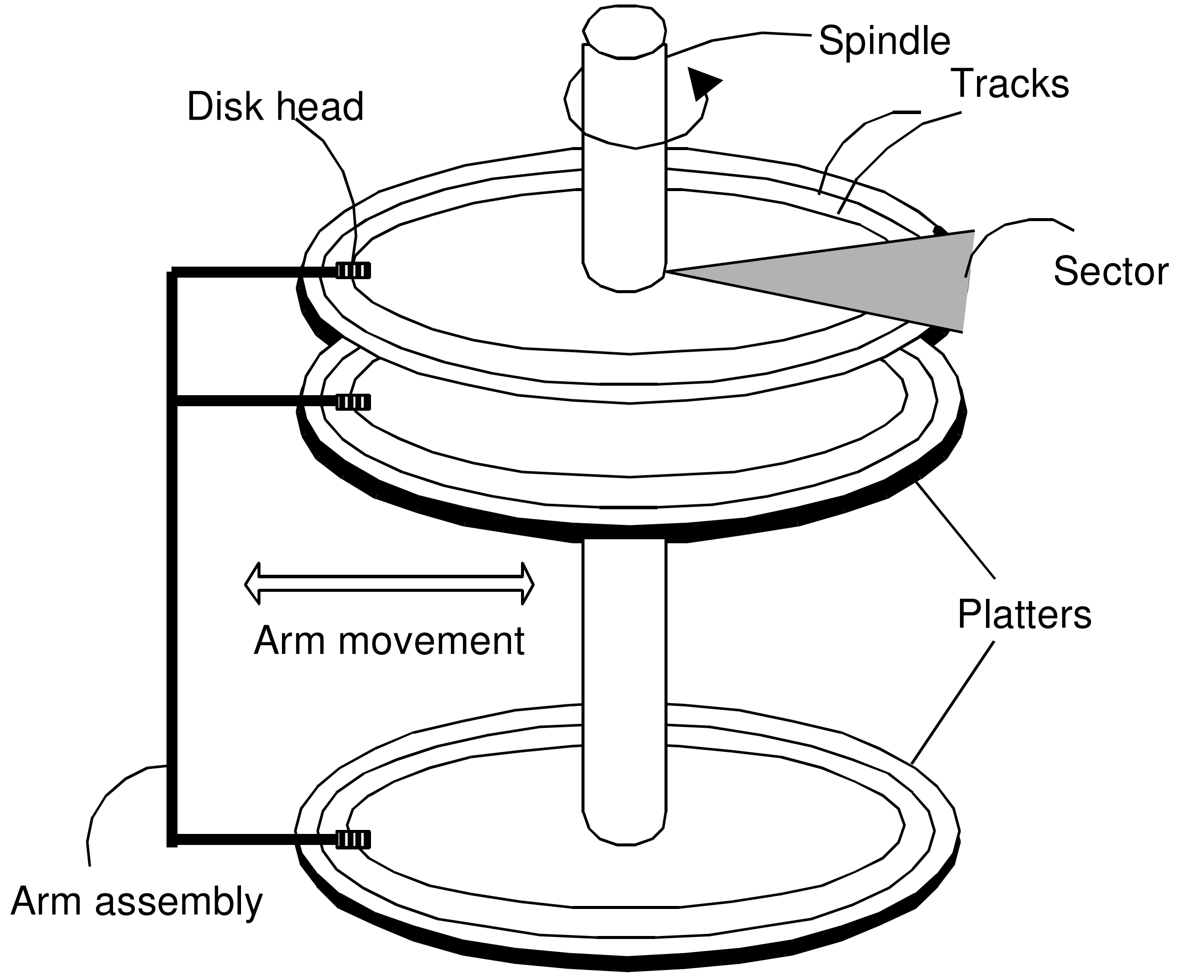
- Bits of data (0’s and 1’s) are stored on circular magnetic platters called disks.
- A disk rotates rapidly (& never stops).
- A disk head reads and writes bits of data as they pass under the head.
- Often, several platters are organized into a disk pack (or disk drive).



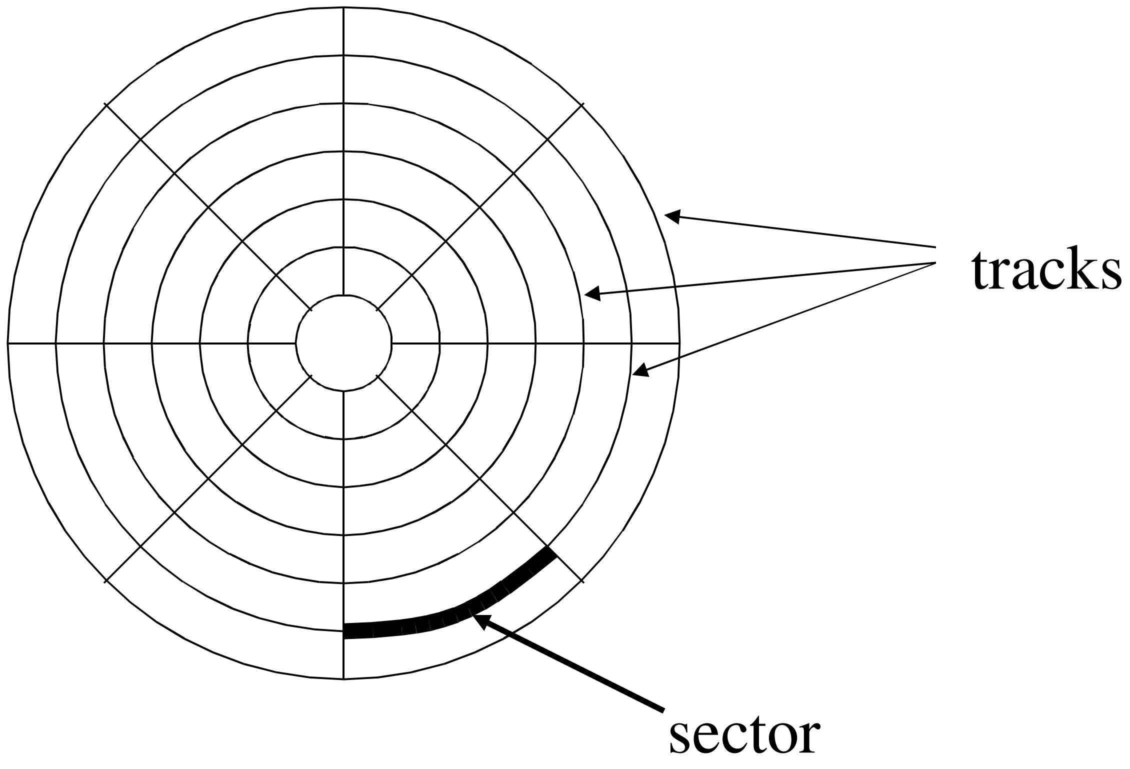
Top view of a 36 GB, 10,000 RPM, IBM SCSI
server hard disk, with its top cover removed.
Note the height of the drive and the 10 stacked platters.
(The IBM Ultrastar 36ZX.)


Cover Mounting Holes (Cover not shown)
Base Casting
Spindle
Slider (and Head)
Actuator Arm
Actuator Axis
Actuator
Case Mounting Holes
Platters
Ribbon Cable (attaches heads to Logic Board)
SCSI Interface Connector
Jumper Pins
Jumper
Power Connector
Tape Seal

Components of a Disk



Looking at a surface



Surface of disk showing tracks and sectors

Organization of Disks

- Disk contains concentric **tracks.**
- Tracks are divided into **sectors**
- A **sector** is the smallest addressable unit in a disk.
- Sectors are addressed by:
  - surface #
  - cylinder (track) #
  - sector #

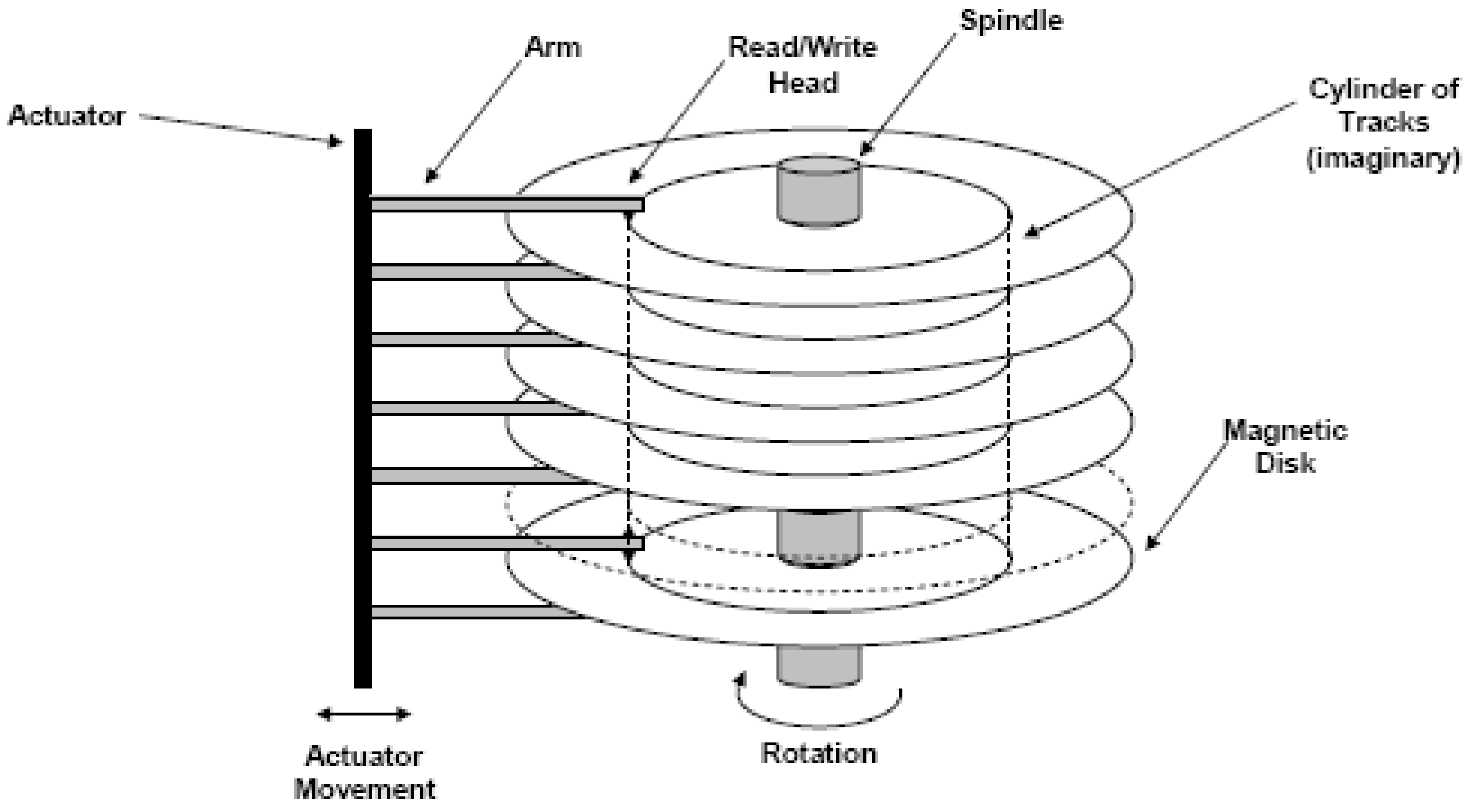
Accessing Data

- When a program reads a byte from the disk, the operating system locates the surface, track and sector containing that byte, and reads the entire sector into a special area in main memory called **buffer**.
- The bottleneck of a disk access is moving the read/write arm. So it makes sense to store a file in tracks that are below/above each other in different surfaces, rather than in several tracks in the same surface.

Cylinders

- A **cylinder** is the set of tracks at a given radius of a disk pack.
  - i.e. a cylinder is the set of tracks that can be accessed without moving the disk arm.
- All the information on a cylinder can be accessed without moving the read/write arm.

Cylinders


Arm
Read/Write Head
Spindle
Actuator
Cylinder of Tracks (imaginary)
Magnetic Disk
Actuator Movement
Rotation

Estimating Capacities

- Track capacity = # of sectors/track * bytes/sector
- Cylinder capacity = # of tracks/cylinder * track capacity
- Drive capacity = # of cylinders * cylinder capacity
- Number of cylinders = # of tracks in a surface

Knowing these relationships allows us to compute the amount of disk space a file is likely to require

Exercise

- Store a file of 20000 records on a disk with the following characteristics:

  # of bytes per sector = 512

  # of sectors per track = 40

  # of tracks per cylinder = 12

  # of cylinders = 1331

Q1. How many cylinders does the file require if each data record requires 256 bytes?

Q2. What is the total capacity of the disk?

Clusters

- Another view of sector organization is the one maintained by the O.S.'s **file manager**.
- It views the file as a series of **clusters** of sectors.
- File manager uses a **file allocation table (FAT)** to map logical sectors of the file to the physical clusters.

Extents

- If there is a lot of room on a disk, it may be possible to make a file consist entirely of contiguous clusters. Then we say that the file is one **extent**. (very good for sequential processing)
- If there isn't enough contiguous space available to contain an entire file, the file is divided into two or more noncontiguous parts. Each part is an extent.

Fragmentation

- ➢ Internal fragmentation: loss of space within a sector or a cluster.
- Due to records not fitting exactly in a sector: **e.g.** Sector size is 512 and record size is 300 bytes. Either
  - store one record per sector, or
  - allow records *span* sectors.
- Due to the use of clusters: If the file size is not a multiple of the cluster size, then the last cluster will be partially used.

Choice of cluster size

- ➢ Some operating systems allow system administrator to choose cluster size.

- When to use large cluster size?

- What about small cluster size?

The Cost of a Disk Access

- The time to access a sector in a track on a surface is divided into 3 components:

Time Component	Action
Seek Time	Time to move the read/write arm to the correct cylinder
Rotational delay (or latency)	Time it takes for the disk to rotate so that the desired sector is under the read/write head
Transfer time	Once the read/write head is positioned over the data, this is the time it takes for transferring data

Seek time

- Seek time is the time required to move the arm to the correct cylinder.
- Largest in cost.

Typically:

- 5 ms (miliseconds) to move from one track to the next (track-to-track)
- 50 ms maximum (from inside track to outside track)
- 30 ms average (from one random track to another random track)

Average Seek Time (***s***)

- Since it is usually impossible to know exactly how many tracks will be traversed in every seek, we usually try to determine the <u>average seek time</u> (***s***) required for a particular file operation.
- If the starting and ending positions for each access are random, it turns out that the average seek traverses one third of the total number of cylinders.
- Manufacturer's specifications for disk drives often list this figure as the average seek time for the drives.
- Most hard disks today have ***s*** of less than 10 ms, and high-performance disks have ***s*** as low as 7.5 ms.

Latency (rotational delay)

- Latency is the time needed for the disk to rotate so the sector we want is under the read/write head.
- Hard disks usually rotate at about 5000rpm, which is one revolution per 12 msec.
- Note:
  - Min latency = 0
  - Max latency = Time for one disk revolution
  - **Average latency (*r*)** = (min + max) / 2
    = max / 2
    = time for ½ disk revolution
- Typically 6 – 8 ms average

Transfer Time

- Transfer time is the time for the read/write head to pass over a block.
- The transfer time is given by the formula:

$$\text{Transfer time} = \frac{\text{number of bytes transferred}}{\text{number of bytes on a track}} \times \text{rotation time}$$

- e.g. if there are 63 sectors per track, the time to transfer one sector would be 1/63 of a revolution.

Exercise

Given the following disk:

- 20 surfaces
  800 tracks/surface
  25 sectors/track
  512 bytes/sector
- 3600 rpm (revolutions per minute)
- 7 ms track-to-track seek time
  28 ms avg. seek time
  50 ms max seek time.

Find:

c) Average latency

d) Disk capacity

e) Time to read the entire disk, one cylinder at a time

Exercise

- Disk characteristics:
  - Average seek time = 8 msec.
  - Average rotational delay = 3 msec
  - Maximum rotational delay = 6 msec.
  - Spindle speed = 10,000 rpm
  - Sectors per track = 170
  - Sector size = 512 bytes
- **Q) What is the average time to read one sector?**

Sequential Reading

- Given the following disk:
  - s = 16 ms
  - r = 8.3 ms
  - Block transfer time = 0.84 ms

b) Calculate the time to read 10 sequential blocks

c) Calculate the time to read 100 sequential blocks

Random Reading

Given the same disk,

b) Calculate the time to read 10 blocks randomly

c) Calculate the time to read 100 blocks randomly

Fast Sequential Reading

- We assume that blocks are arranged so that there is no rotational delay in transferring from one track to another within the same cylinder. This is possible if consecutive track beginnings are staggered (like running races on circular race tracks)
- We also assume that the consecutive blocks are arranged so that when the next block is on an adjacent cylinder, there is no rotational delay after the arm is moved to new cylinder
- *Fast sequential reading*: no rotational delay after finding the first block.

Consequently …

Reading b blocks:

ii. Sequentially:

$$\underbrace{s + r}_{\text{insignificant for large files}} + b * \text{btt}$$

$$\Rightarrow \; b * \text{btt}$$

vi. Randomly:

$$b * (s + r + \text{btt})$$

Exercise

- Given a file of 30000 records, 1600 bytes each, and block size 2400 bytes, how does record placement affect sequential reading time?

ii) Empty space in blocks.

iv) Records overlap block boundaries.

Exercise

- **Specifications of a 300MB disk drive**:
  - Min seek time = 6ms.
  - Average seek time = 18ms
  - Rotational delay = 8.3ms
  - transfer rate = 16.7 ms/track or 1229 bytes/ms
  - Bytes per sector = 512
  - Sectors per track = 40
  - Tracks per cylinder = 12
  - Tracks per surface = 1331
  - Interleave factor = 1
  - Cluster size= 8 sectors
  - Smallest extent size = 5 clusters

Q) How long will it take to read a 2048Kb file that is divided into 8000 256 byte records?

i) **Access the file sequentially**

ii) **Access the file randomly**

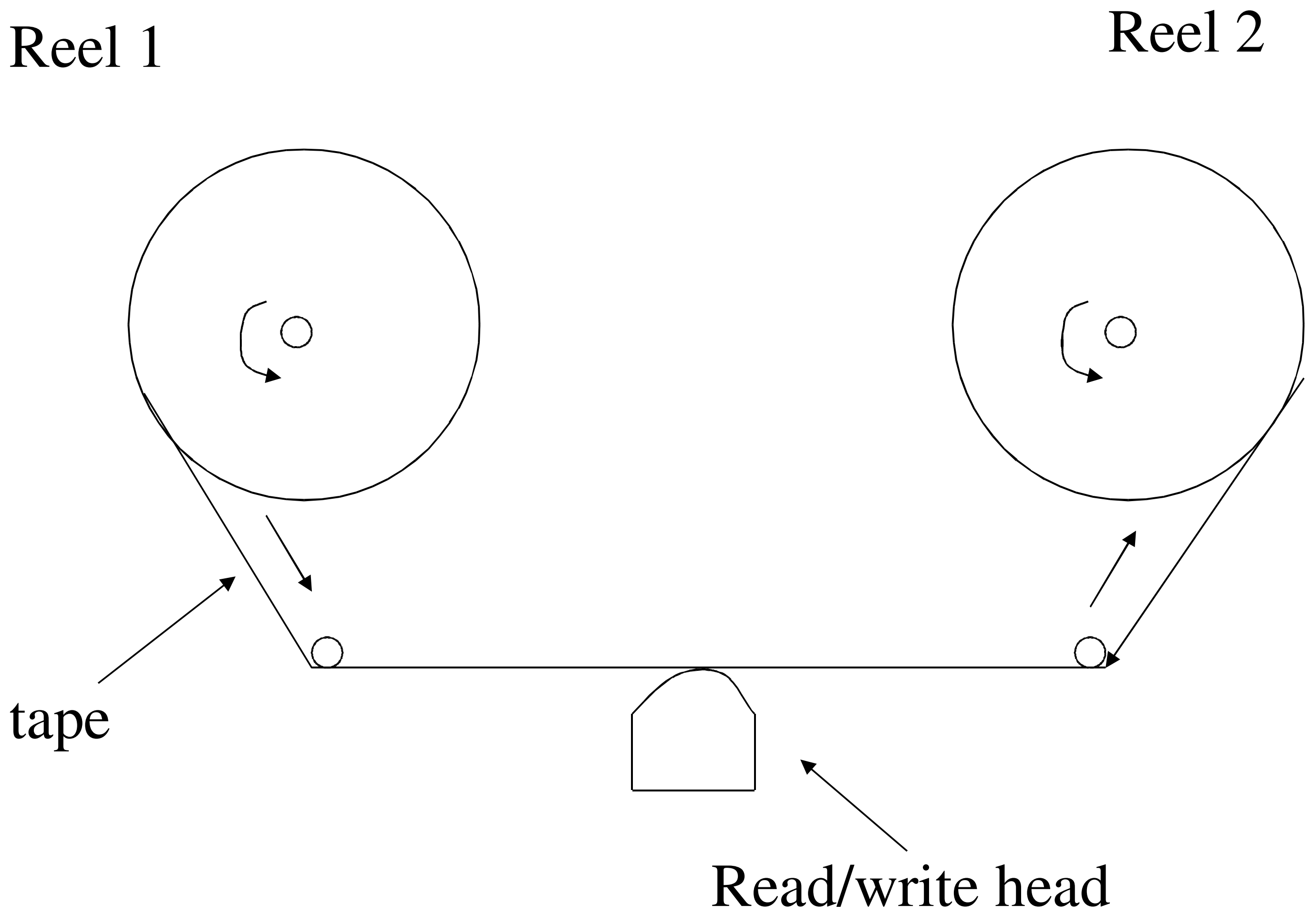
Secondary Storage Devices: Magnetic Tapes

Characteristics

- No direct access, but very fast sequential access.
- Resistant to different environmental conditions.
- Easy to transport, store, cheaper than disk.
- Before it was widely used to store application data; nowadays, it’s mostly used for backups or archives

Magnetic tapes

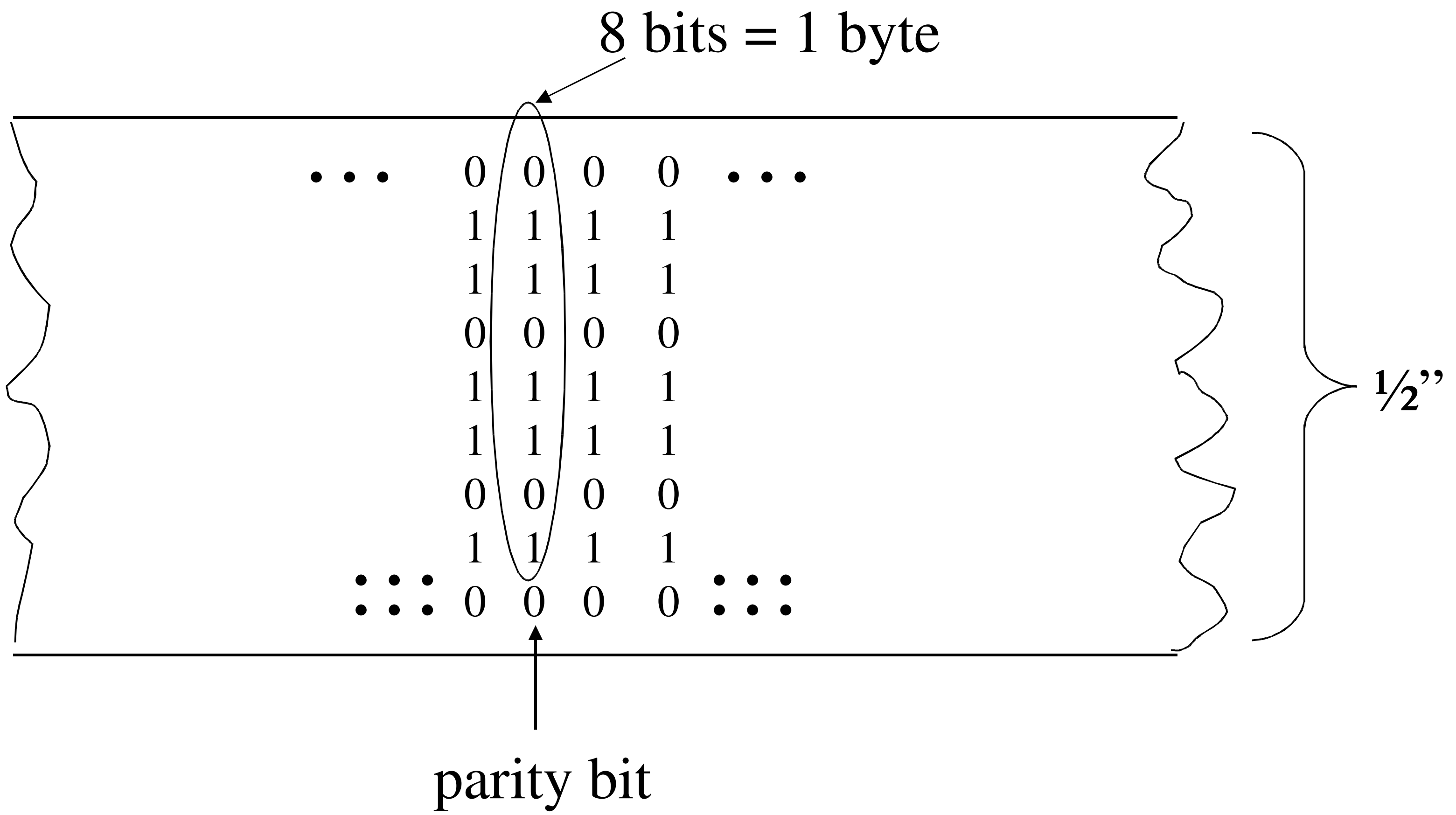
- A sequence of bits are stored on magnetic tape.
- For storage, the tape is wound on a reel.
- To access the data, the tape is unwound from one reel to another.
- As the tape passes the head, bits of data are read from or written onto the tape.


Reel 1
Reel 2
tape
Read/write head

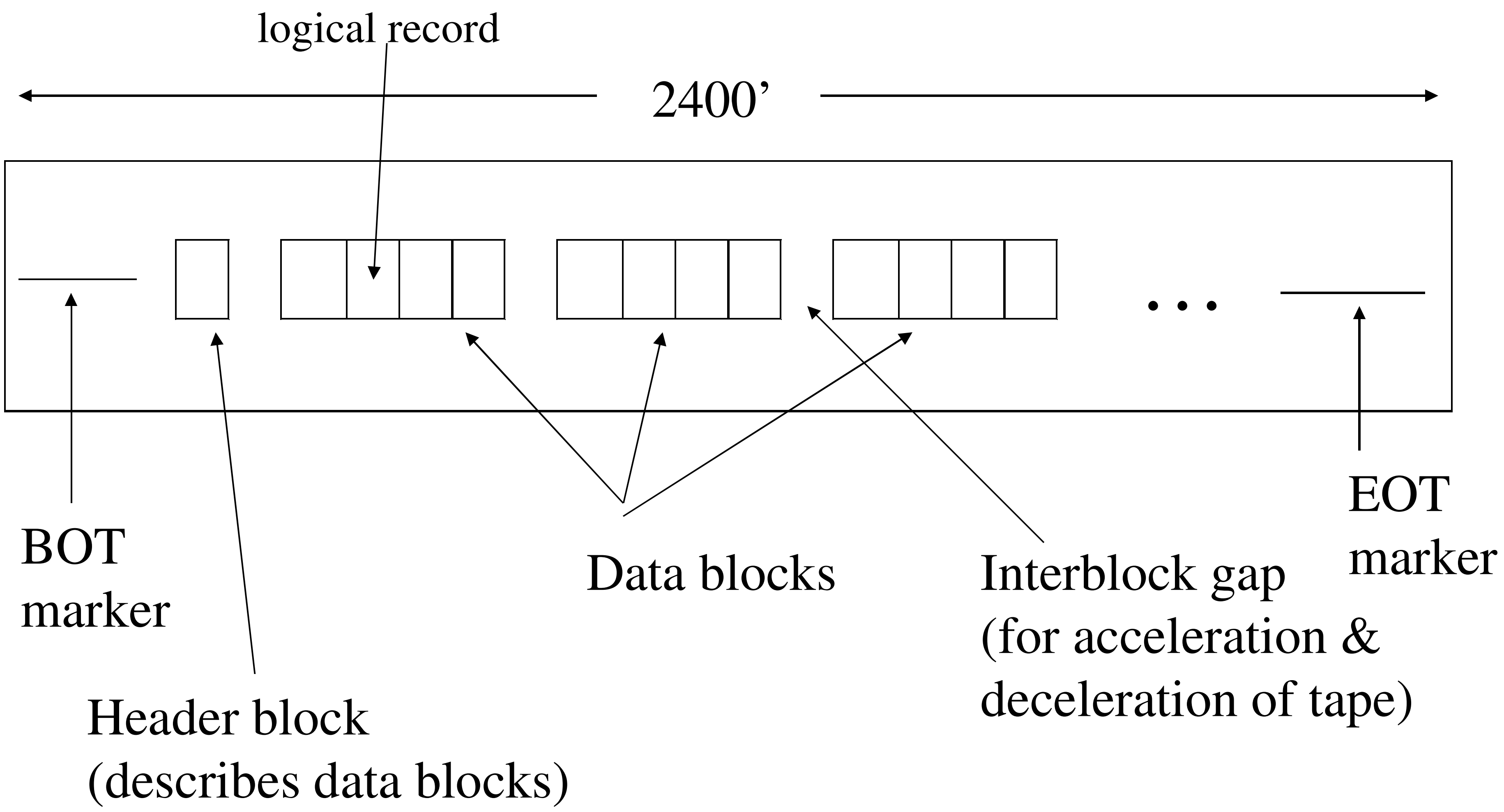
Tracks

- Typically data on tape is stored in 9 separate bit streams, or <u>tracks</u>.
- Each track is a sequence of bits.
- Recording density = # of bits per inch (bpi).
  Typically 800 or 1600 bpi.
  30000 bpi on some recent devices.

In detail


8 bits = 1 byte
0 0 0 0
1 1 1 1
1 1 1 1
0 0 0 0
1 1 1 1
1 1 1 1
0 0 0 0
1 1 1 1
0 0 0 0
½"
parity bit

Tape Organization


logical record
2400’
. . .
BOT marker
Header block (describes data blocks)
Data blocks
Interblock gap (for acceleration & deceleration of tape)
EOT marker

Data Blocks and Records

- Each data block is a sequence of contiguous records.
- A record is the unit of data that a user's program deals with.
- The tape drive reads an entire block of records at once.
- Unlike a disk, a tape starts and stops.
- When stopped, the read/write head is over an interblock gap.

Example: tape capacity

- Given the following tape:
  - Recording density = 1600 bpi
  - Tape length = 2400 '
  - Interblockgap = ½ "
  - 512 bytes per record
  - Blocking factor = 25
- How many records can we write on the tape? (ignoring BOT and EOT markers and the header block for simplicity)

Secondary Storage Devices: CD-ROM

Physical Organization of CD-ROM

- Compact Disk – read only memory (write once)
- Data is encoded and read optically with a laser
- Can store around 600MB data
- Digital data is represented as a series of **Pits** and **Lands:**
  - Pit = a little depression, forming a lower level in the track
  - Land = the flat part between pits, or the upper levels in the track

Organization of data

- Reading a CD is done by shining a laser at the disc and detecting changing reflections patterns.
  - 1 = change in height (land to pit or pit to land)
  - 0 = a “fixed” amount of time between 1’s

```
  LAND     PIT    LAND          PIT    LAND
...-------+     +-------------+       +---...
          |_____|             |_______|
..0 0 0 0 1 0 0 1 0 0 0 0 0 0 1 0 0 0 1 0 0 ..
```

- Note : we cannot have two 1’s in a row!
  => uses Eight to Fourteen Modulation (EFM) encoding table.

Properties

- Note that: Since 0's are represented by the **length of time** between transitions, we must travel at **constant linear velocity** (**CLV**)on the tracks.
- Sectors are organized along a spiral
- Sectors have same linear length
- Advantage: takes advantage of all storage space available.
- Disadvantage: has to change rotational speed when seeking (slower towards the outside)

Addressing

- 1 second of play time is divided up into 75 **sectors**.
- Each sector holds 2KB
- 60 min CD:
  60min * 60 sec/min * 75 sectors/sec =
  270,000 sectors = 540,000 KB ~ 540 MB
- A **sector** is addressed by:
  Minute:Second:Sector
  e.g. 16:22:34

A journey of a Byte and Buffer Management

Reference: Sections 3.8 & 3.9

A journey of a byte

- Suppose in our program we wrote:

```
outfile << c;
```

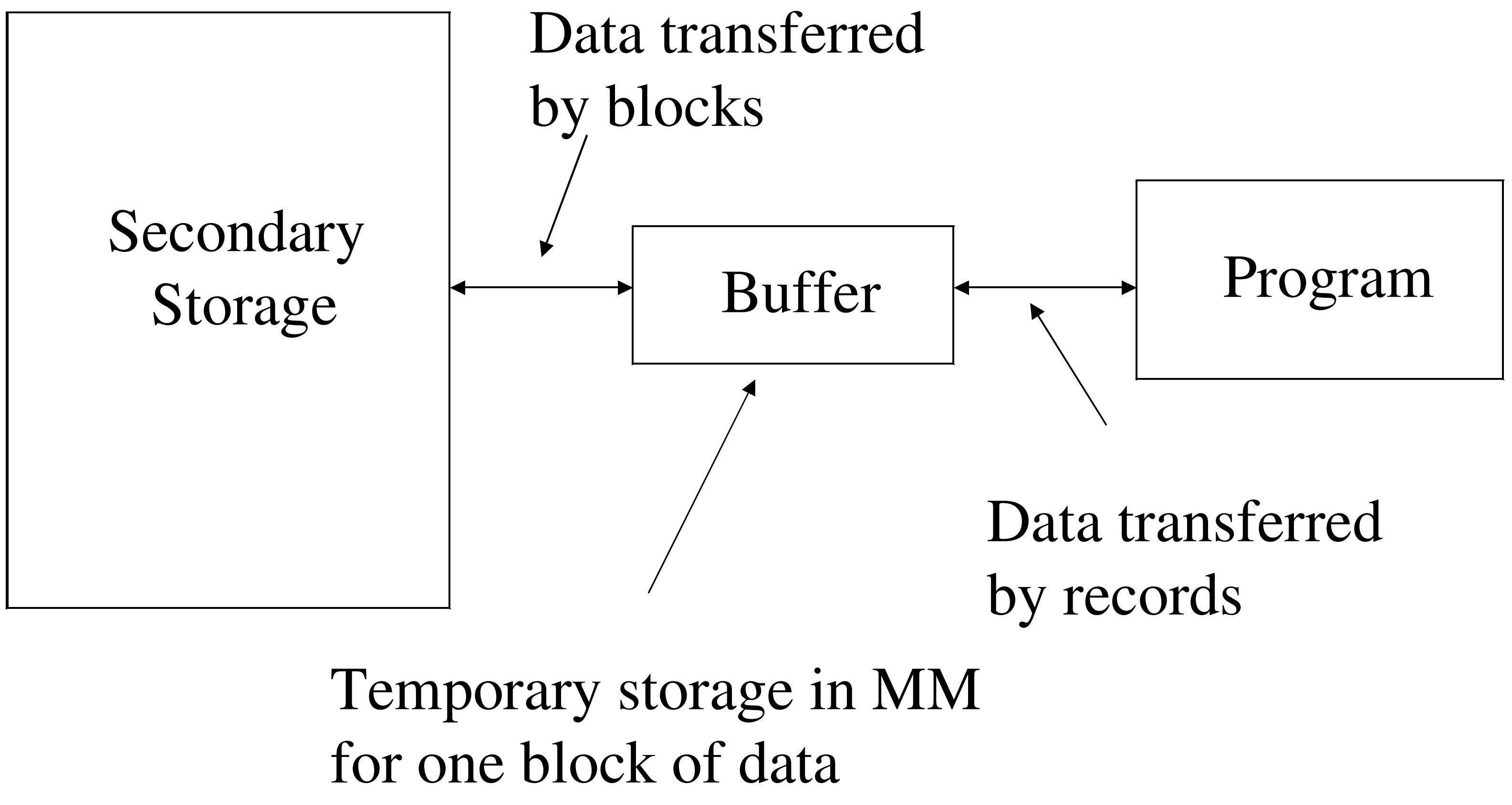
- This causes a call to the **file manager** (a part of O.S. responsible for I/O operations)
- The O/S (File manager) makes sure that the byte is written to the disk.
- Pieces of software/hardware involved in I/O:
  - Application Program
  - Operating System/ file manager
  - I/O Processor
  - Disk Controller

- **Application program**
  - Requests the I/O operation
- **Operating system / file manager**
  - Keeps tables for all opened files
  - Brings appropriate sector to buffer.
  - Writes byte to buffer
  - Gives instruction to I/O processor to write data from this buffer into correct place in disk.
  - Note: the buffer is an exact image of a cluster in disk.
- **I/O Processor**
  - a separate chip; runs independently of CPU
  - Find a time when drive is available to receive data and put data in proper format for the disk
  - Sends data to disk controller
- **Disk controller**
  - A separate chip; instructs the drive to move R/W head
  - Sends the byte to the surface when the proper sector comes under R/W head.

Buffer Management

- Buffering means working with large chunks of data in main memory so the number of accesses to secondary storage is reduced.
- Today, we’ll discuss the System I/O buffers. These are beyond the control of application programs and are manipulated by the O.S.
- Note that the application program may implement its own “buffer” – i.e. a place in memory (variable, object) that accumulates large chunks of data to be later written to disk as a chunk.
- Read Section 4.2 for using classes to manipulate program buffers.

System I/O Buffer


Data transferred by blocks
Secondary Storage
Buffer
Program
Data transferred by records
Temporary storage in MM for one block of data

Buffer Bottlenecks

- Consider the following program segment:

```
while (1) {
  infile >> ch;
    if (infile.fail()) break;
    outfile << ch;
}
```

- What happens if the O.S. used only one I/O buffer?

  ⇒ Buffer bottleneck

- Most O.S. have an input buffer and an output buffer.

Buffering Strategies

- **Double Buffering:** Two buffers can be used to allow processing and I/O to overlap.
  - Suppose that a program is only writing to a disk.
  - CPU wants to fill a buffer at the same time that I/O is being performed.
  - If two buffers are used and I/O-CPU overlapping is permitted, CPU can be filling one buffer while the other buffer is being transmitted to disk.
  - When both tasks are finished, the roles of the buffers can be exchanged.
- The actual management is done by the O.S.

Other Buffering Strategies

- Multiple Buffering: instead of two buffers any number of buffers can be used to allow processing and I/O to overlap.
- Buffer pooling:
  - There is a pool of buffers.
  - When a request for a sector is received, O.S. first looks to see that sector is in some buffer.
  - If not there, it brings the sector to some free buffer. If no free buffer exists, it must choose an occupied buffer. (usually LRU strategy is used)