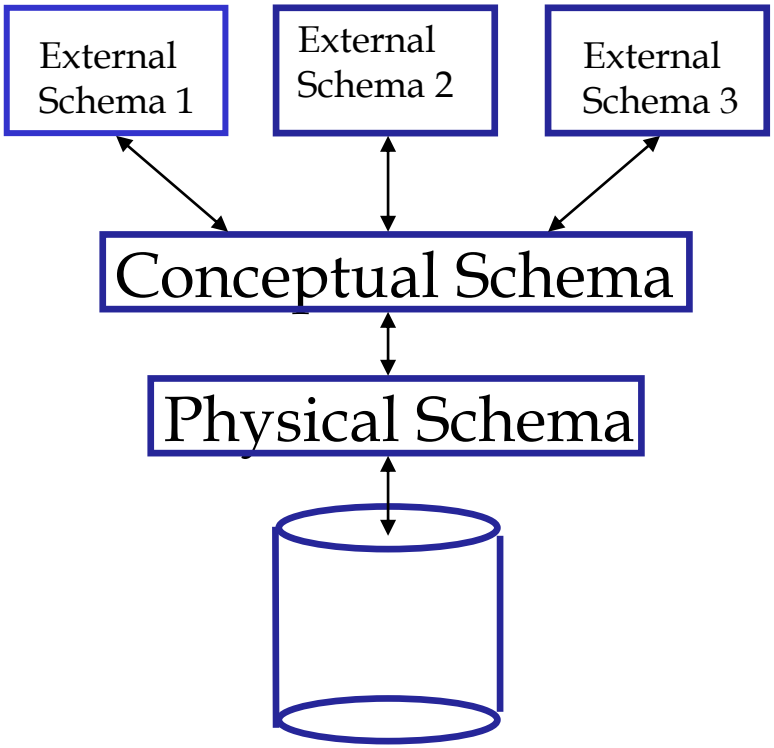
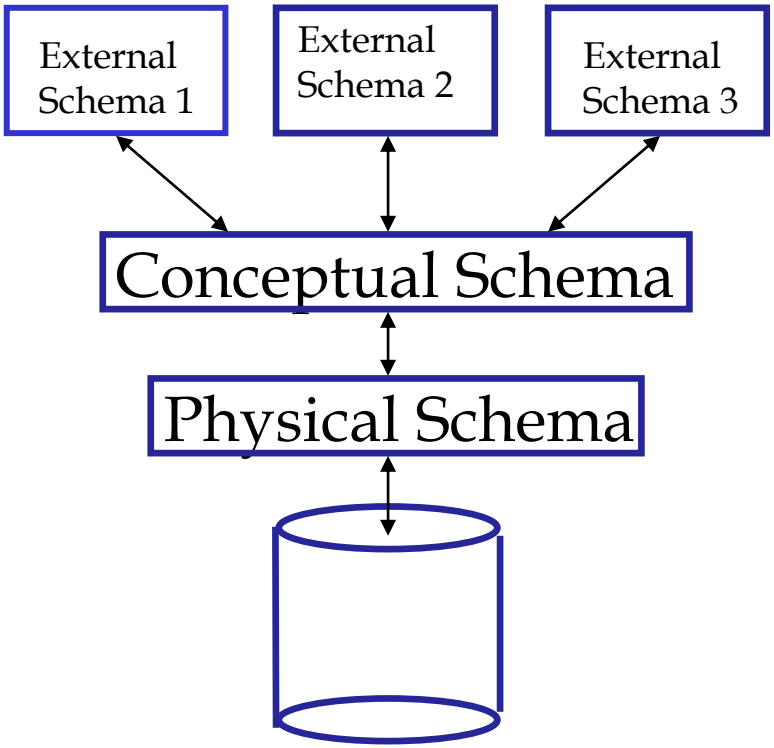
Secondary Storage Devices

Remember the Levels of Abstraction



Remember the Levels of Abstraction

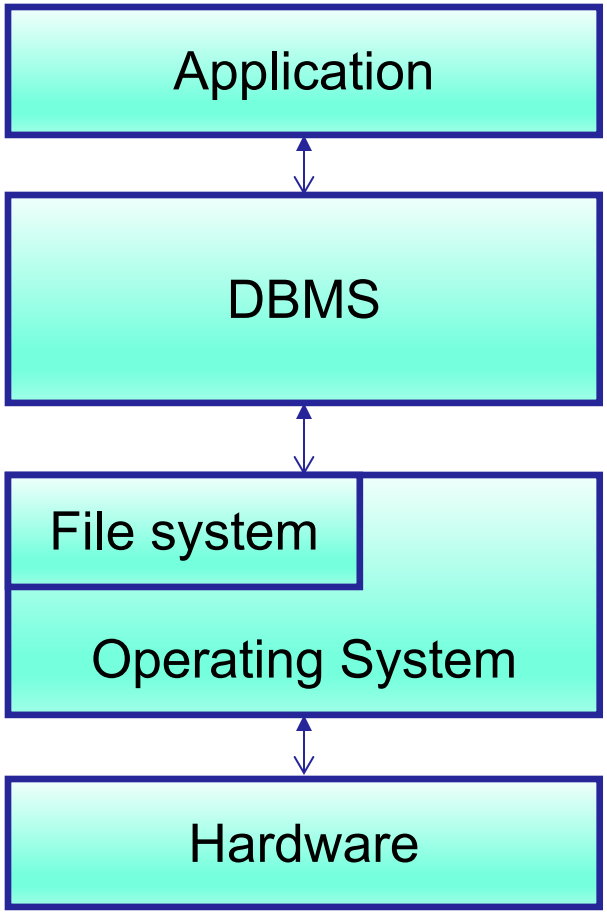


- So far we studied on the Conceptual Schema
- We start focusing on the Physical Schema

File Structures

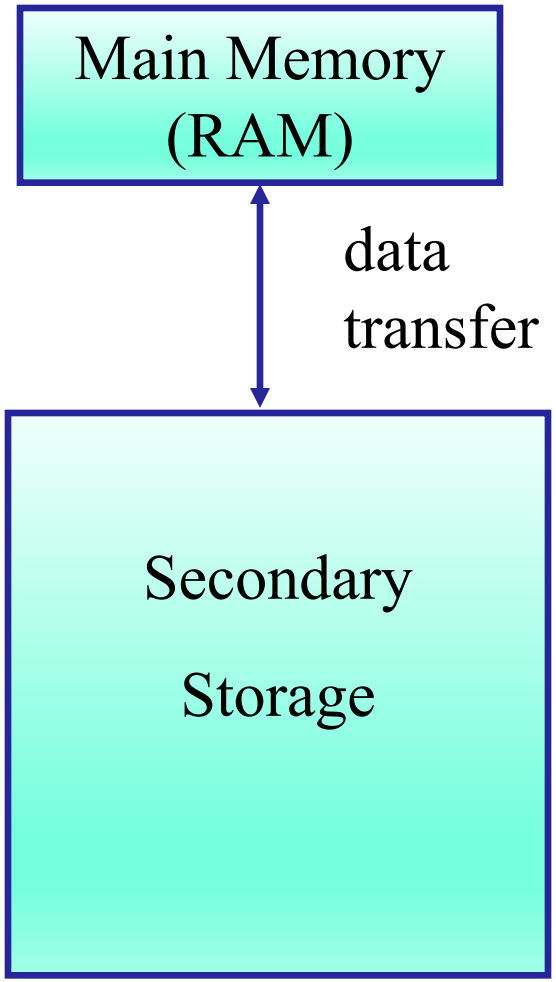
- How are the database tables stored on disk?
- We will study file processing in databases:
  - Storage of data
  - Organization of data
  - Access to data
  - Processing of data

Where do File Structures fit in Computing?



Computer Architecture

data is manipulated here



- Semiconductors
- Fast, expensive, volatile, small

data is stored here

- Disks (HDD, SSD), tape
- Slow, cheap, stable, large

How fast is main memory?

- Typical time for getting info from:

  Main memory: ~10 nanosec = 10 x 10^{-9} sec

  Magnetic disks:

  ~5-10 milisec (HD) = 10 x 10^{-3} sec

  ~25-100 microsec (SSD) = 100 x 10^{-6} sec
- Keeping same time proportion as above:

  MM: 1 sec

  SSD: 2.7 hours

  HD: 11.57 days

Goal of the file structures

- Minimize the number of trips to the disk in order to get desired information
- Grouping related information so that we are likely to get everything we need with only one trip to the disk.

File Systems

- Data is not scattered hither and thither on disk.
- Instead, it is organized into **files**.
- Files are organized into **records**.
- Records are organized into **fields**.

Example

- A Student file may be a collection of student records, one record for each student
- Each student record may have several fields, such as
  - Student id
  - Student name
  - Major
  - Semester
  - Age
  - Gender
  - Address
  - …
- Typically, each record in a file has the same fields.

Secondary Storage Devices

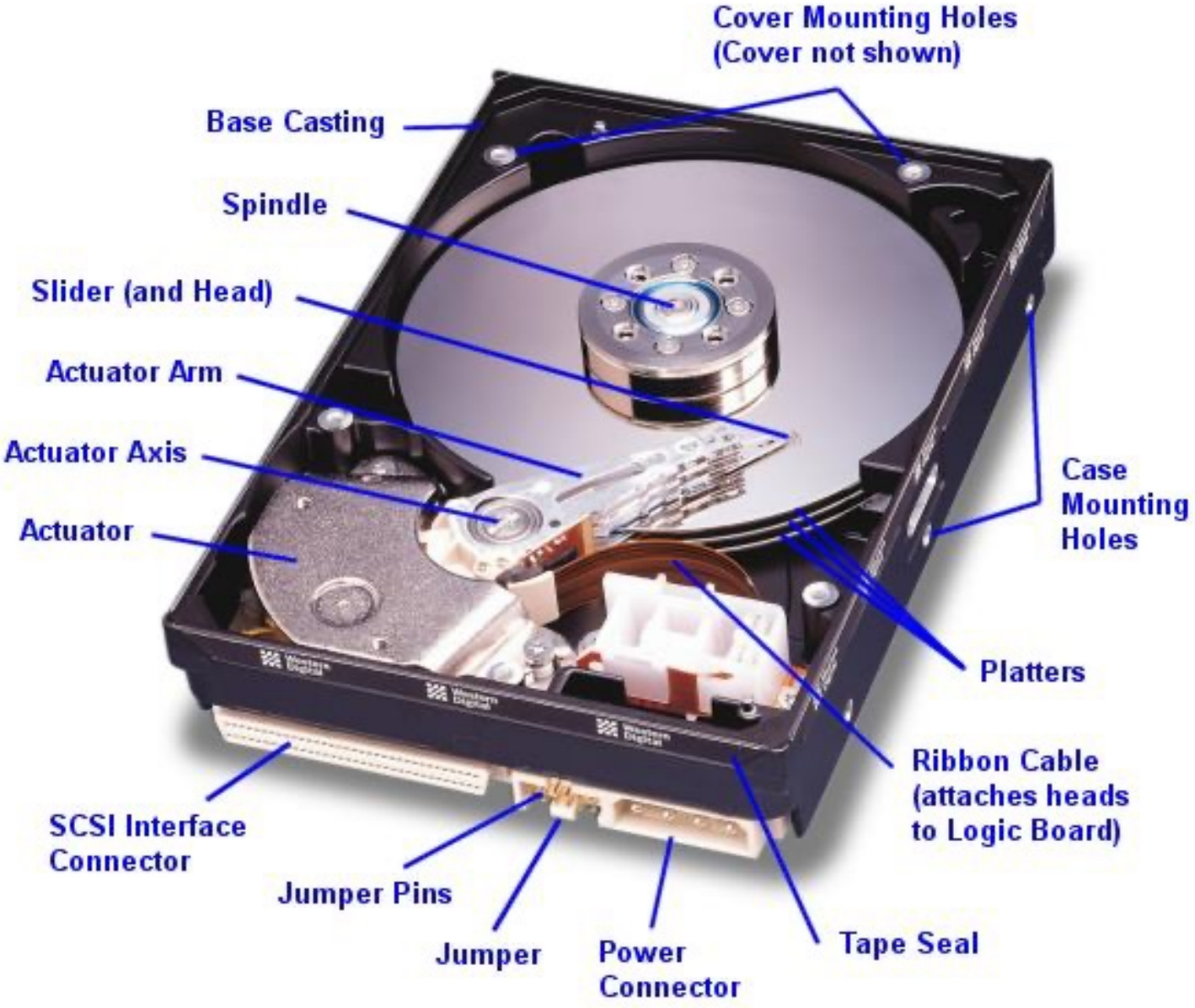
Secondary Storage Devices

- A secondary storage device refers to any volatile storage device that is internal or external to the computer.
  - Magnetic Disks- Hard disk drives (high capacity, low cost per bit)
  - Solid state drives (SSD)
  - Optical Disks
    CD (Compact disc)
    DVD

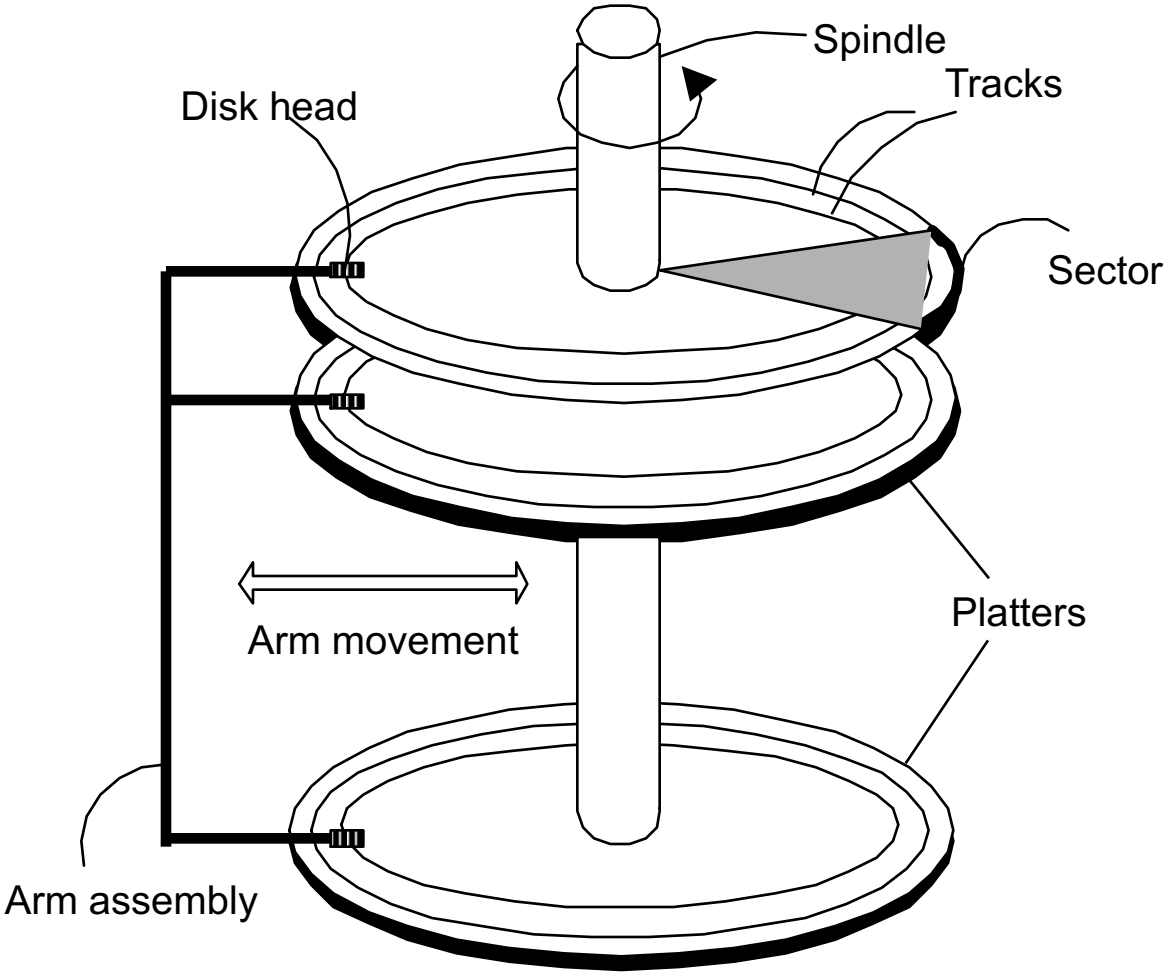
Magnetic Disks

- 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).

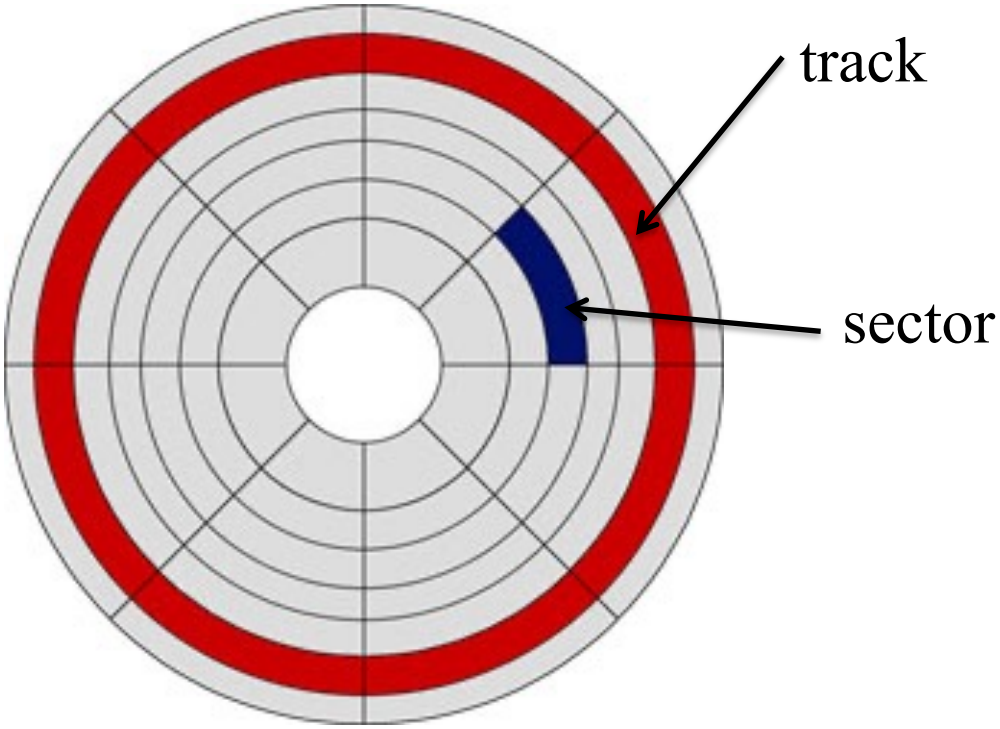
Top view of a 36 GB, 10,000 RPM, IBM SCSI
server hard disk, with its top cover removed.
Note the height of the drive and the 10 stacked platters.
(The IBM Ultrastar 36ZX.)


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 of one platter



Surface of disk showing tracks and sectors

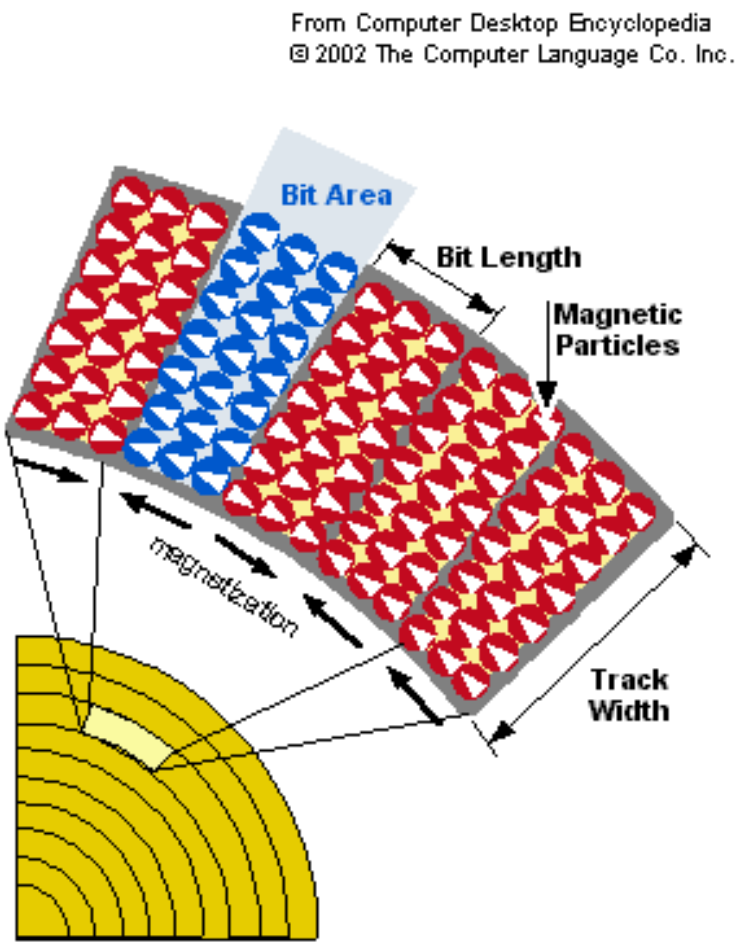
https://www.youtube.com/watch?v=Cj8-WNjaGuM

https://www.youtube.com/watch?v=wteUW2sL7bc

https://www.youtube.com/watch?v=kdmLvl1n82U

How hard drives hold data

- Magnetizable plates with magnet points arranged in sectors and tracks.
- These positively and negatively magnetized points define the 0s and 1s.
- Hard Drive's head is a kind of electromagnet that convert polarization into electronic signals.
- When writing data, the reverse happens, and the electronic signals are magnetized back into the platter via the head.



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 #

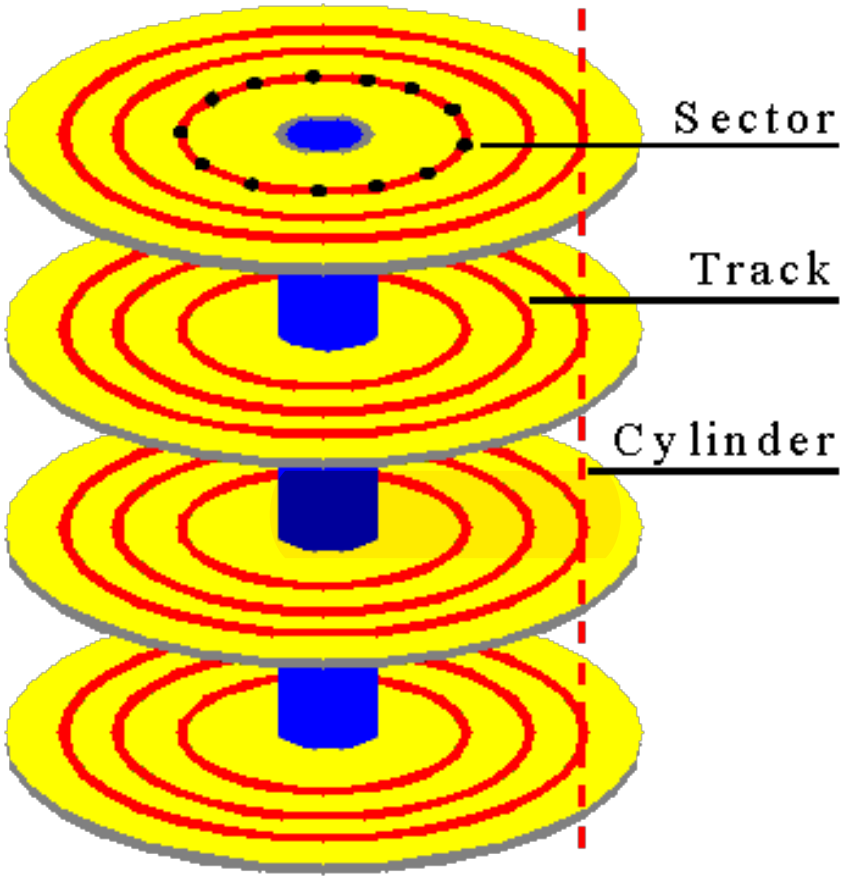
Accessing Data

- 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



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. What is the total capacity of the disk?

Q2. How many cylinders does the file require if each data record requires 256 bytes?

- 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. What is the total capacity of the disk?

512*40*12*1331 = 327106560 B

311 GB

- 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

Q2. How many cylinders does the file require if each data record requires 256 bytes?

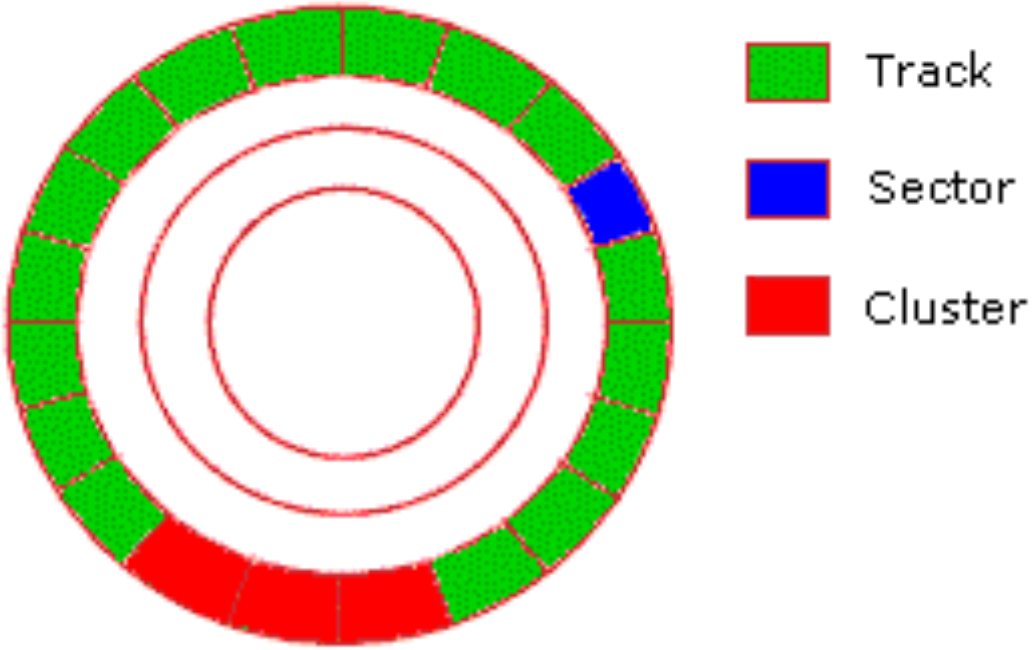
20000*256 / (512*40*12)

Clusters

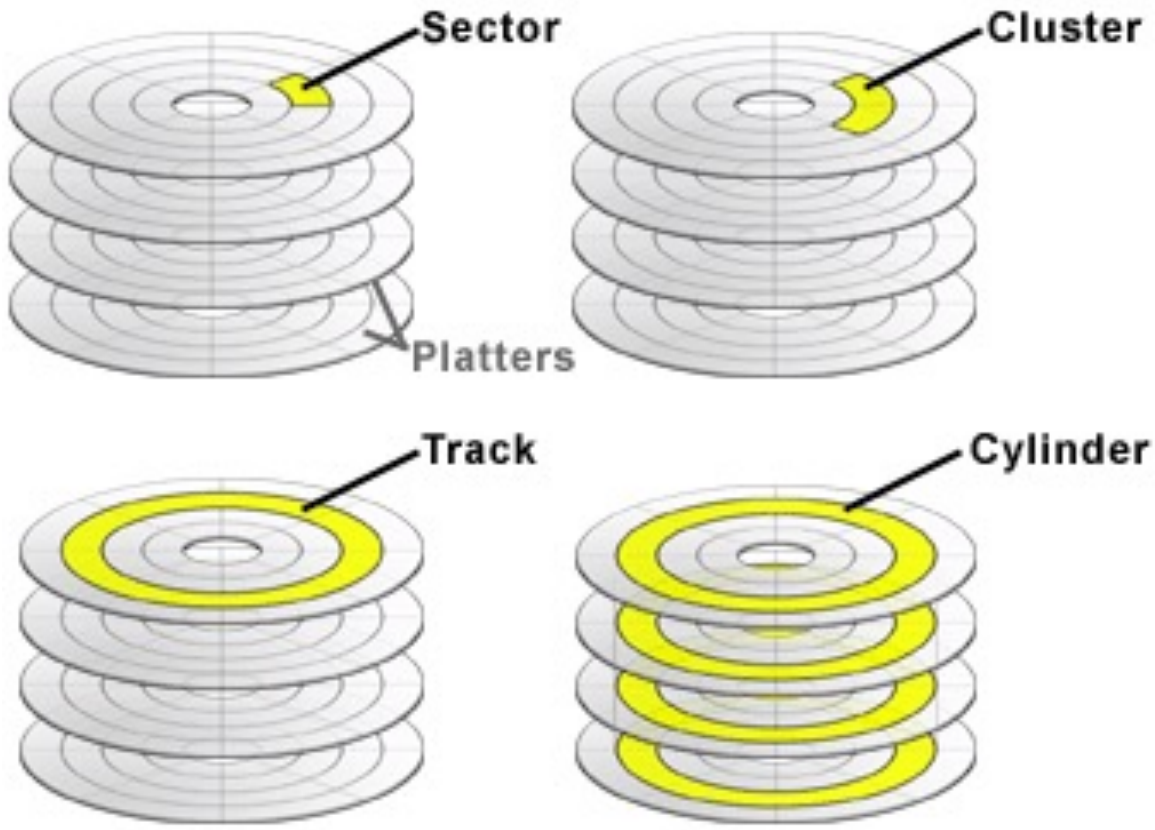
- Another view of sector organization is the one maintained by the Operating System's **file manager**.
- A cluster is the smallest unit of storage space with which the Operating System will deal.
- O.S. views the file as a series of **clusters** of sectors.
- File manager uses a **file allocation table (FAT)** to map logical sectors of the file to the physical clusters.

→ Note that in DBMS, the term "***block***" is used to denote the data storage/transfer unit

Cluster



Organization on disk



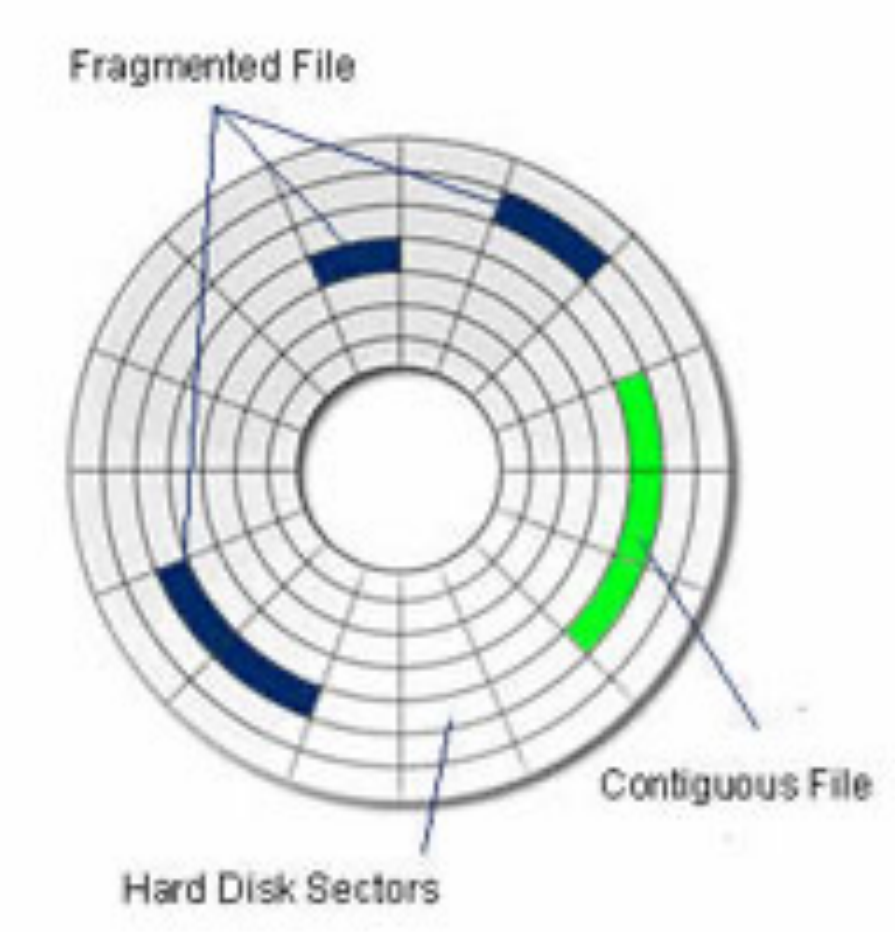
Extents

- If there is a lot of room on a disk, it may be possible to make a file consist entirely of contiguous blocks/sectors. Then we say that the file is *one* piece. (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 pieces.
- Each piece is called an **extent**.

File Fragmentation

- Individual file fragmentation occurs when a single file has been broken into multiple extents.
- File system fragmentation increases disk head movement or seeks, which are known to hinder throughput.

Fragmentation



Internal Fragmentation

Internal fragmentation happens due to the use of blocks/sectors:

- If the file size is not a multiple of the block size, then the last block will be partially used.

Choice of cluster/block size

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

- When to use large cluster/block size?

- What about small cluster/block 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 **average seek time (*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*** less than 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 7200rpm, which is one revolution per 8.33 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 3 – 7 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:

a) Average latency
b) Disk capacity
c) Time to read the entire disk, one cylinder at a time

Exercise

Given the following disk:

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

Find:

a) Average latency

Exercise

Given the following disk:

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

Find:

b) Disk capacity

Exercise

Given the following disk:

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

Find:

c) Time to read the entire disk,

one cylinder at a time

Solution

a) Average Latency:

3600 rev/min => 60 rev/sec

$\Rightarrow$ 1/60 sec/rev = 0.0167 sec = 16.7 ms

$\Rightarrow$ Average latency = r = 16.7/2 = 8.3 ms

b) Disk capacity

25*512*800*20 = 204.8MB

c) Time to read the disk:

Track read time = 1 revolution time= 16.7 ms

Cylinder read time = 20*16.7= 334ms

Total read time = 800*cylinder reads + 799 cylinder switches

= 800*334 ms + 799 * 7ms

= 267 sec + 5.59 sec = 272.59 sec

Fast Sequential Reading

- We assume that sectors 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 sectors are arranged so that when the next sector 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 sector.

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?**

Solution

- Average time to read one sector:

  s + r + btt

- What is btt?

  btt : block transfer time = revolution time/ #of sectors per track

  Revolution time = 60/10000 = 0.006 sec

  btt = 0.006/170 = 0.035 ms

- s + r + btt = 8+3+0.035 =11.035 ms

Sequential Reading

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

a) Calculate the time to read 10 sequential sectors

b) Calculate the time to read 100 sequential sectors

Solution

a) Reading 10 sequential sectors:

= s + r+ 10 * btt

= 16 + 8.3 + 10 * 0.84 = 32.7 ms

b) 100 sectors:

= 16 + 8.3 + 100 * 0.84 = 108.3 ms

Random Reading

Given the same disk,

a) Calculate the time to read 10 sectors randomly

b) Calculate the time to read 100 sectors randomly

Solution

a) Reading 10 sectors randomly:

$= 10 * (s + r + btt)$

$= 10 * (16 + 8.3 + 0.84) = 251.4$ ms

b) 100 sectors:

$= 100 *(16 + 8.3 + 0.84) = 2514$ ms

Consequently …

Reading *b* sectors:

i. Sequentially:

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

$$\approx b * btt$$

ii. Randomly:

$$b * (s + r + btt)$$

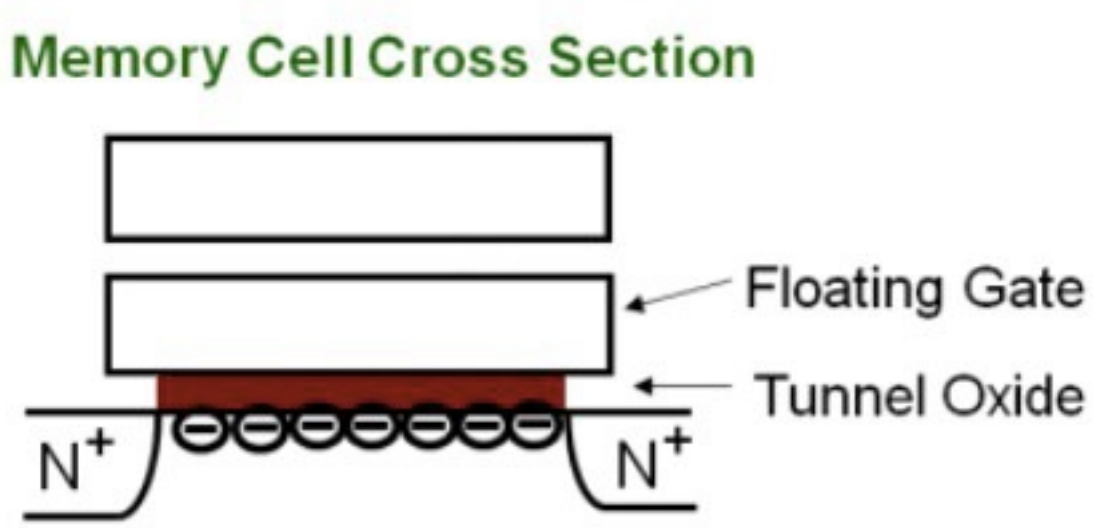
Secondary Storage Devices: Solid State Disk

Solid State Drive (SSD)

- NAND memory cell
- permanently store charge
- programming puts electrons on floating gate
- erase takes them off

SSD

- NAND memory cell
- permanently store charge
- programming puts electrons on floating gate
- erase takes them off



https://www.youtube.com/watch?v=jr1w44m3bWA

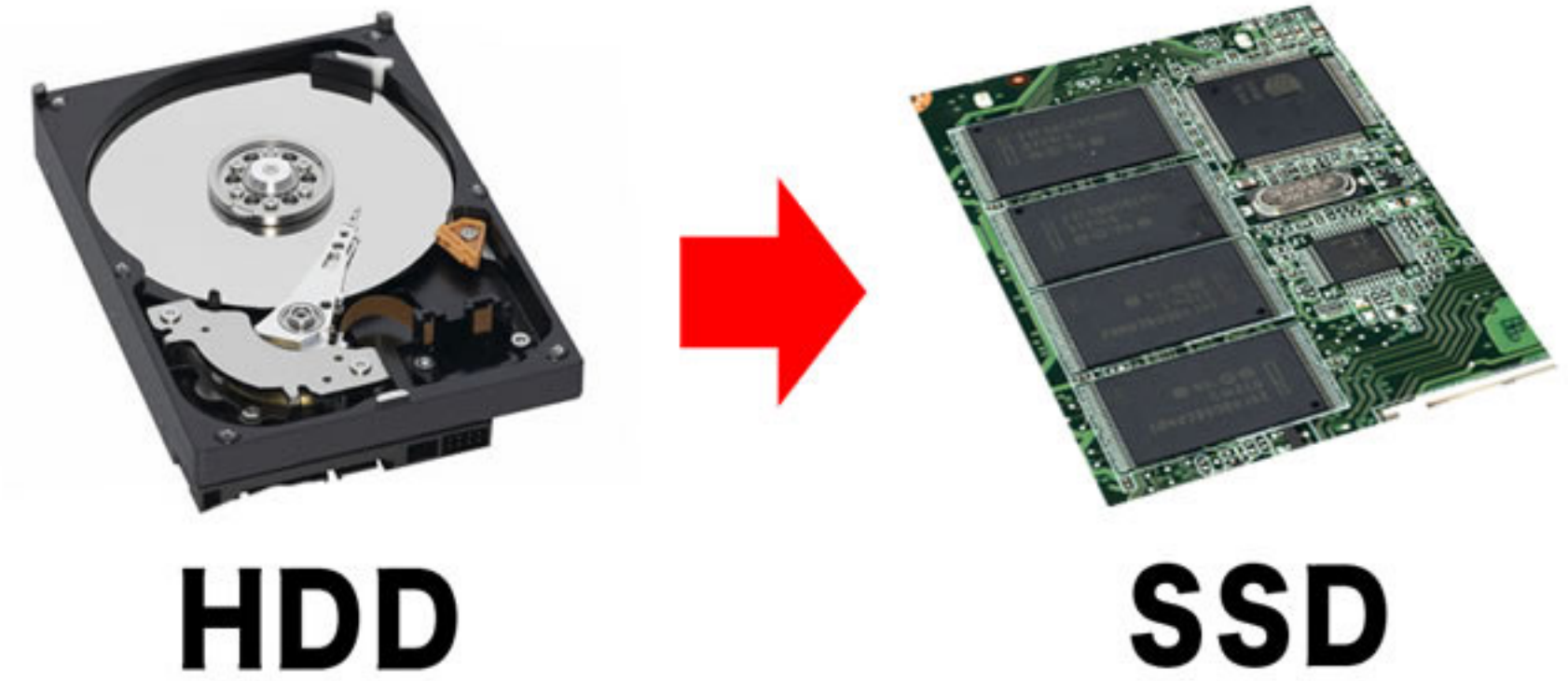
Advantages

- Reliability in portable environments and no noise
  - No moving parts
- Faster start up
  - Does not need spin up
- Extremely low read latency
  - No seek time (25 μs per page/4KB)
- Deterministic read performance
  - The performance does not depend on the location of data

Disadvantage

- Cost significantly more per unit capacity
  - 3$/GB vs. 0.15$/GB
- Limited write erase time
- Slower write speeds because of the erase blocks are becoming larger and larger (1.5 ms per erase)
- For low capacity flash SSDs, low power consumption and heat production when in active use. High capacity SSDs may have significant higher power requirements

SSD layout



SSD Layout

1) memory cells
2) pages & blocks
3) planes
4) dies
5) TSOPs & SSDs

SSD layout

- pages: multiple memory cells
  - one page is the smallest structure which can be read or written
- blocks: multiple pages
  - one block is the smallest structure which can be erased
  - e.g. one block = 128 pages each 4 KB

    → 512 KB Block
- planes
  - multiple blocks make up a plane
  - e.g. 1.024 Blocks = 1 Plane → 512 MB
- multiple planes make up a die,
  - e.g. 4 Planes = 1 Die → 2 GB

SSD layout

- **TSOPs (thin small outline packages)**
  - multiple dies make up a TSOP
  - typically one – two dies in a TSOP
  - up to eight dies possible
    → 64 GiByte in a TSOP

Source: Intel/Micron

- **SSDs**
  - multiple TSOPs (e.g. ten)
    make up a SSD
  - currently capacities up to 600 GB

SSD layout

- An example layout
  - 1 die = 4 planes
  - 1 plane = 2048 blocks
  - 1 block = 64 pages
  - 1 page = 4KB
- Dies can operate independently
- Reading and programming is performed on a page basis, erasure can only be performed on a block basis.

Typical read and write rates

	Drive Model	Description	Seek Time			Latency	Read XFR Rate		Write XFR Rate	
			Track to Track	Average	Full Stroke		Outer Tracks	Inner Tracks	Outer Tracks	Inner Tracks
Hard Drives	Western Digital WD7500AYYS	7200 RPM 3.5" SATA	0.6 ms	8.9 ms	12.0 ms	4.2 ms	85 MB/sec	60 MB/sec*	85 MB/sec	60 MB/sec*
	Seagate ST936751SS	15K RPM 2.5" SAS	0.2 ms	2.9 ms	5.0 ms*	2.0 ms	112 MB/sec	79 MB/sec	112 MB/sec	79 MB/sec
Flash SSDs	Transcend TS8GCF266	8GB 266x CF Card	0.09ms				40 MB/sec		32 MB/sec	
	Samsung MCAQE32G5APP	32G 2.5" PATA	0.14ms				51 MB/sec		28 MB/sec	
	Sandisk SATA5000	32G 2.5" SATA	0.125ms				68 MB/sec		40 MB/sec	

Buffer Management

A journey for Disk Access

- Suppose the following command is sent:

  UPDATE Student S
  SET S.cgpa=3.70
  WHERE S.studentID = 1234;

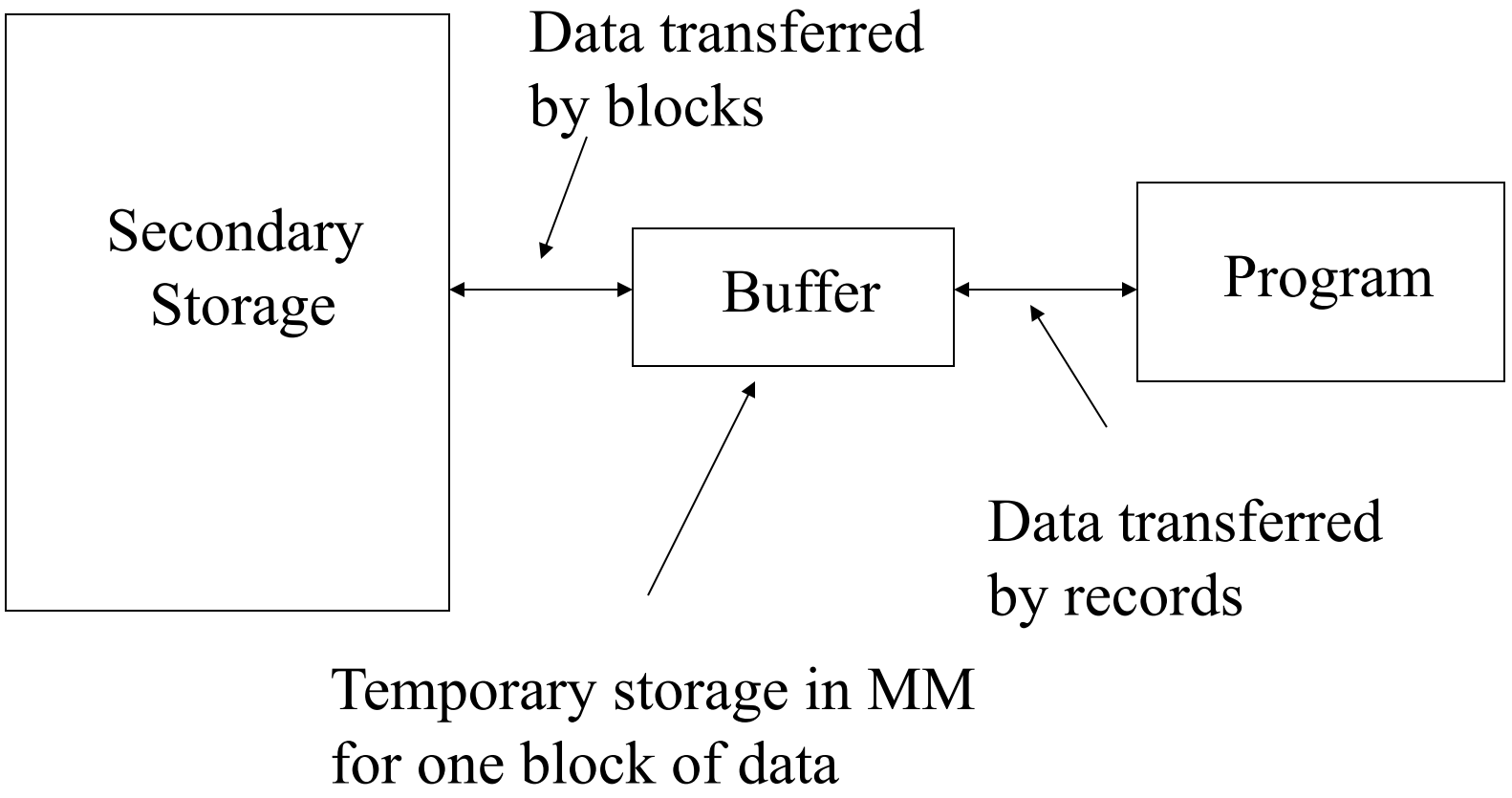
- This causes a call to the **file manager** (a part of O.S. responsible for I/O operations)
- The O/S (File manager) makes sure that the update is written to the disk.
- Pieces of software/hardware involved in I/O:
  - Application Program
  - Operating System/ file manager
  - I/O Processor
  - Disk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

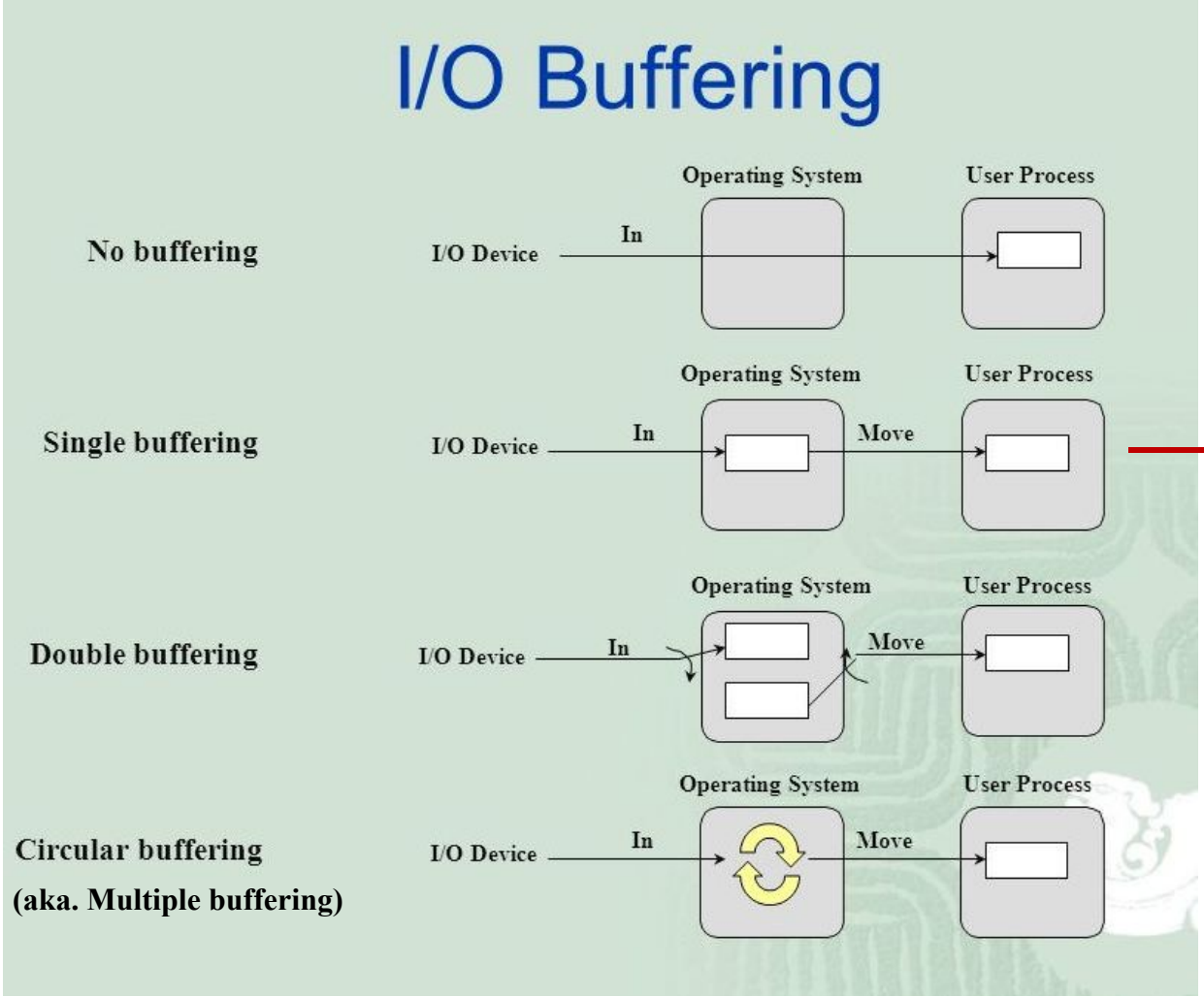
Buffer Management

- Buffering means working with large chunks of data in main memory so the number of accesses to secondary storage is reduced.
- Buffer: Reserved area in the main storage to hold the transferred data to/ from secondary storage. (System I/O buffers)
- 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.

Buffer


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

Buffering



Source: https://www.ques10.com/p/25210/discuss-input-output-buffer-in-detail-1/

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)