Database Management Systems

Introduction to Data Management and File Structure

Topics

- Introduction to File Systems
- Problems in using Disks
- Disk Structure
- Disk I/O Timing Parameters
- Definitions
  - Blocks and Records
  - Buckets
  - Double Buffering
  - Blocking Factor
- Solid State Drive (SSD)

Information Systems

- Many computer systems need to store a large amount of data.
- Examples are: Student information system, Hospital information system, etc.
- This information cannot be stored in computer memory because
  - Memory has a limited capacity
  - Information is lost when we turn off the computer (Volatile memory)

How to store data?

- Data is stored in files.
- Files are stored on disks because disks:
  - Have larger capacity
  - Can store data even when we turn off the computer (non-volatile)

Problems in using disks

- Hard disks are very slow

  Typical time to read an integer from

  RAM = 60 nsec

  Disk = 6 msec

  RAM is 10 million times faster

How to speed up I/O from a disk?

- For faster input/output we can organize data of the files.
- File structure aim is to develop file formats for faster input/output operations

Example: Sorting files

Using Indexes

Hashing

Sorting Files

Advantage:

Search in a sorted file is faster (binary search)

Disadvantage:

Keeping file sorted is difficult (insert, update)

Indexing

- Index is a list, showing the location of records in data files
- For faster indexing, trees are used (example B+tree)

Hashing

- Hashing refers to methods for finding the location of records in data files
- Hashing is faster than indexing

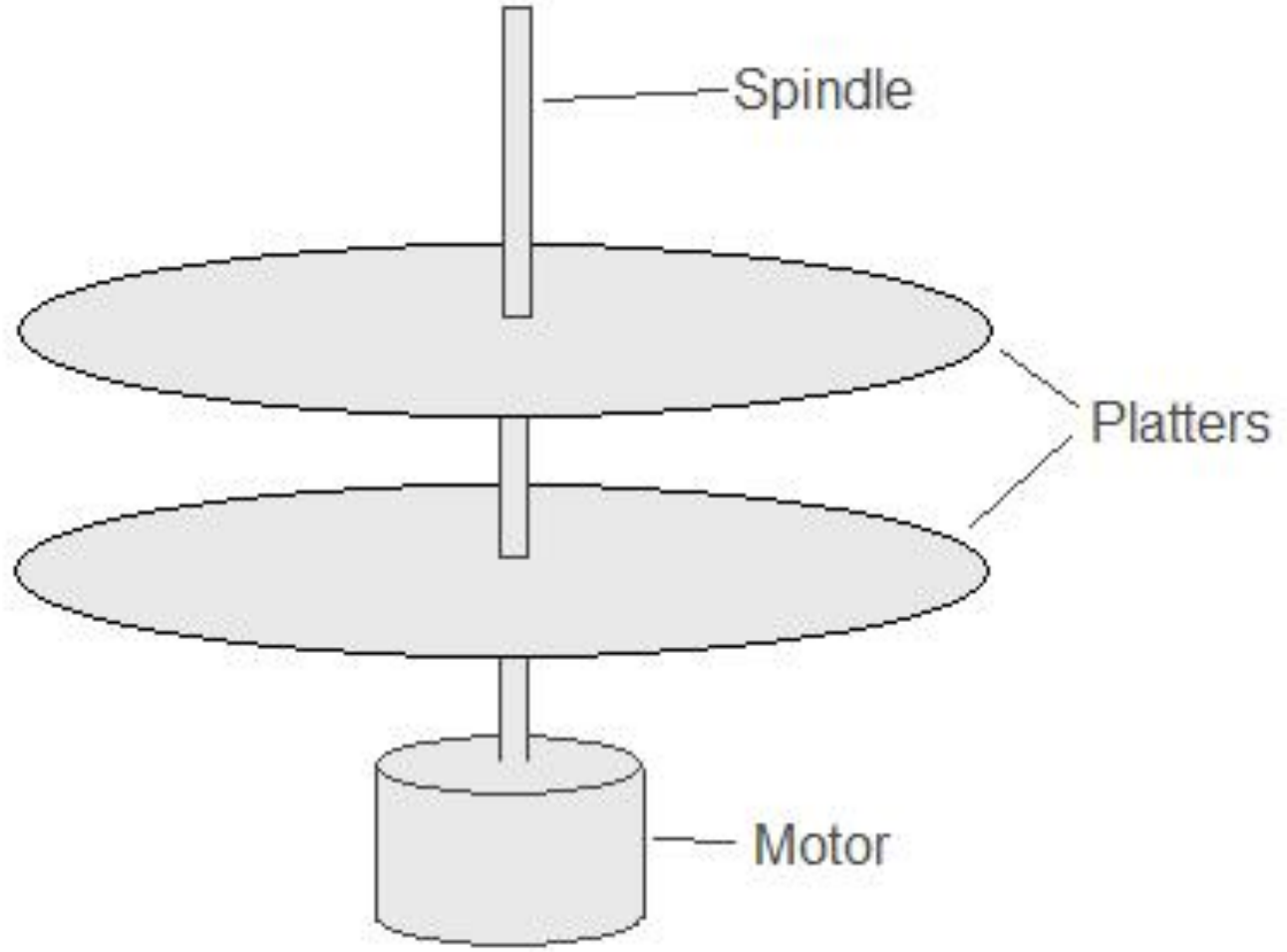
Disks

- Disks are slow compared to RAM
- Disk I/O can be optimized by organizing file data.

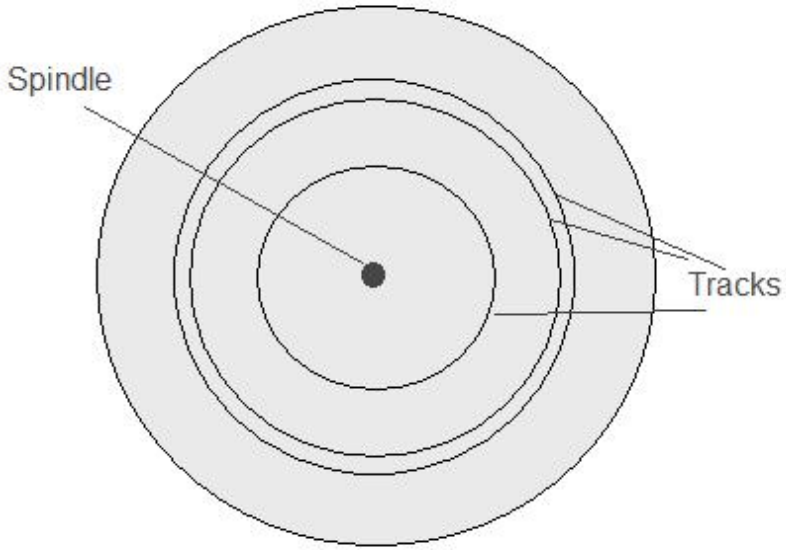
Disk Structure

- Disks have
  - platters to store data
  - spindle to hold platters
  - Motor to turn spindle and hence platters
  - Head to read/write data
  - Arm to hold and move head

Disk Structure

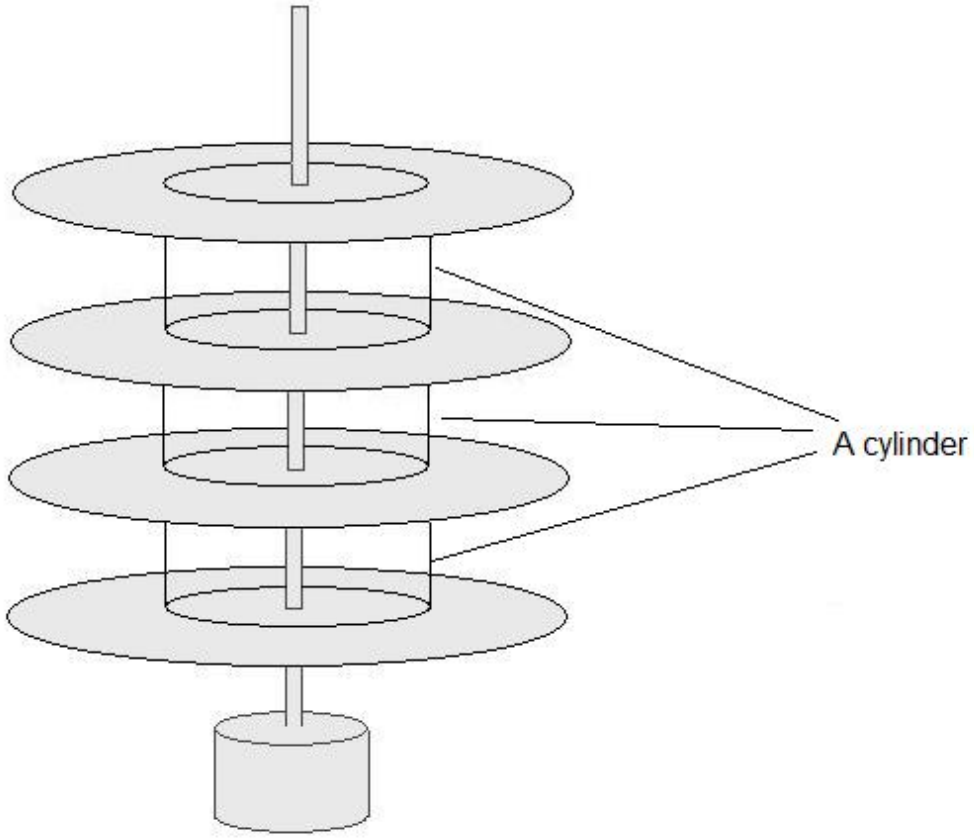


Tracks

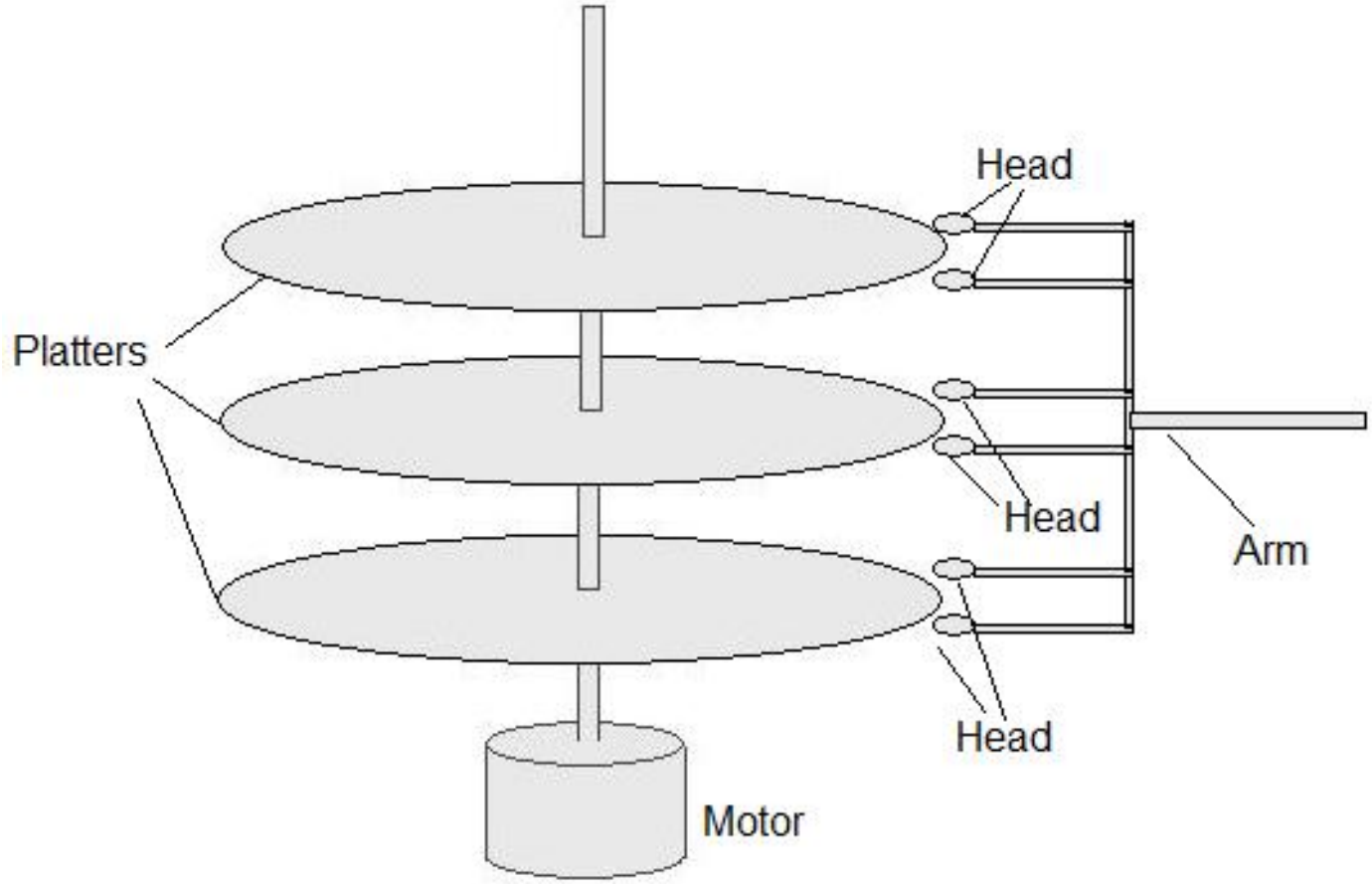


Cylinder

- Tracks of different platters with the same distance from the center (spindle)

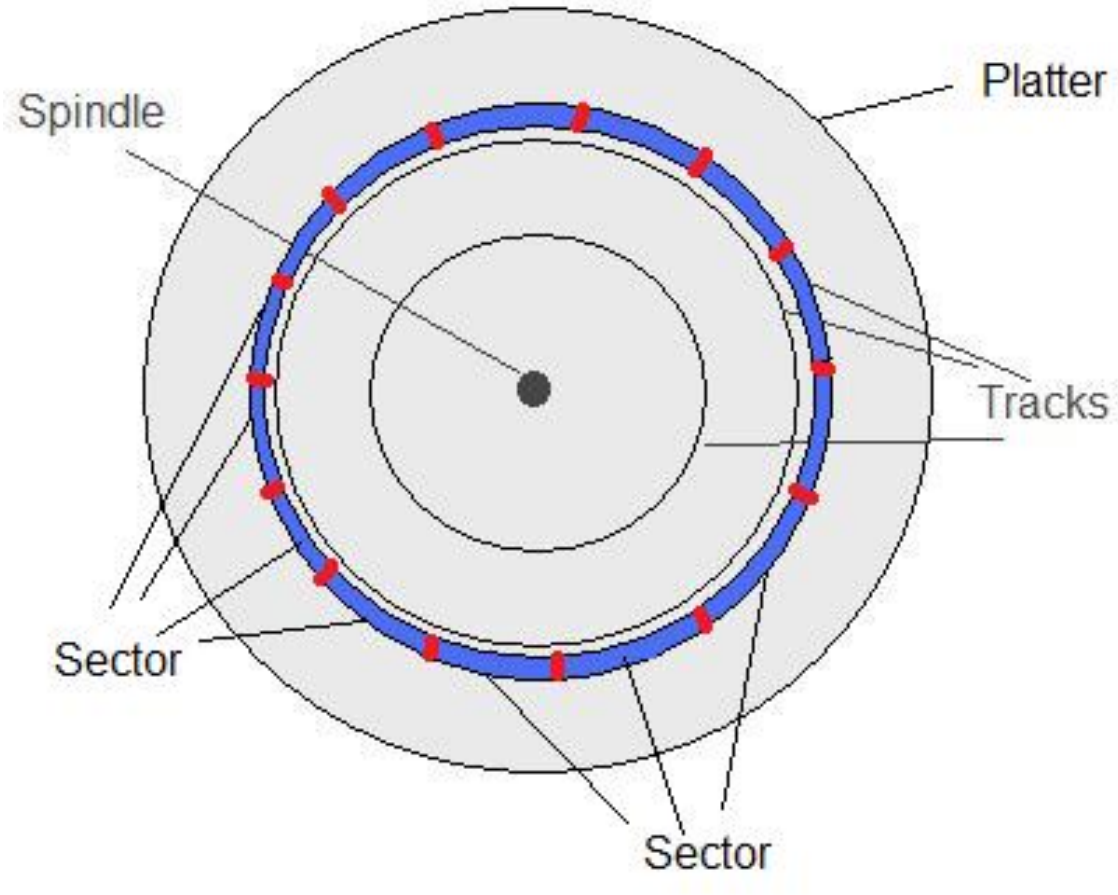


Heads and Arm



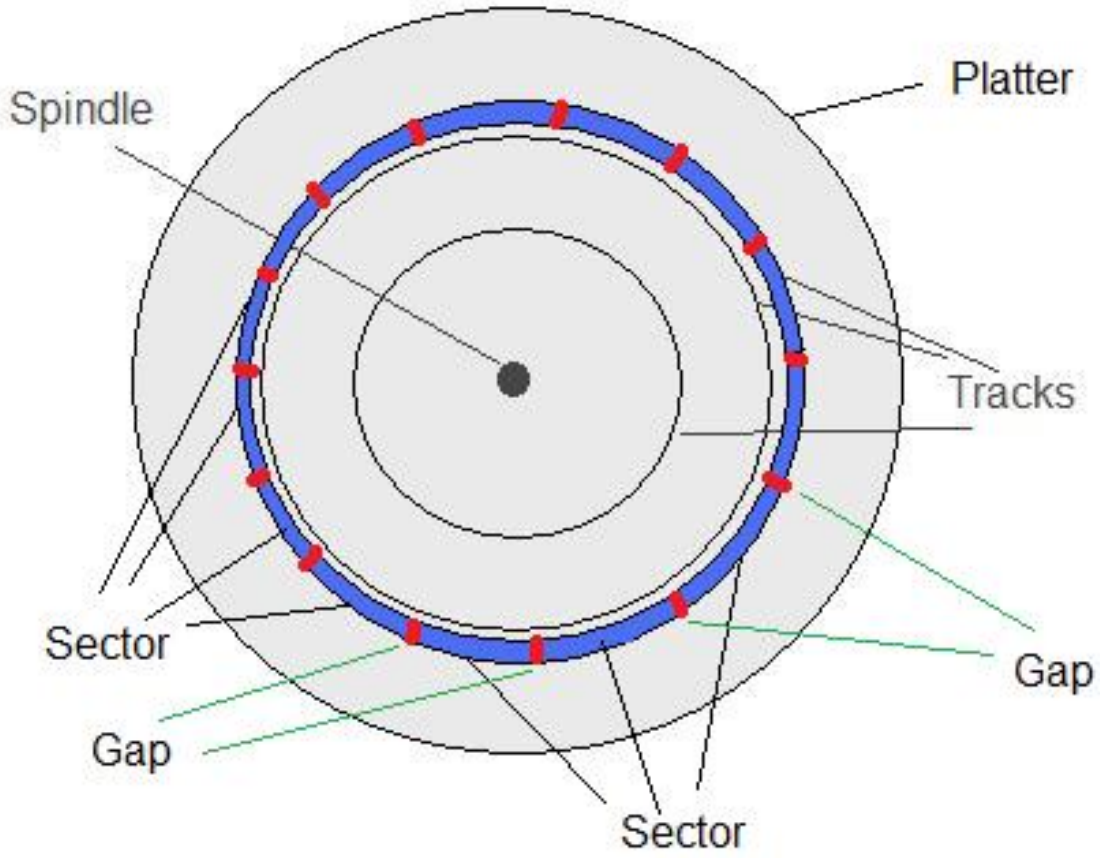
Sectors

- Each track is divided into smaller parts called sectors



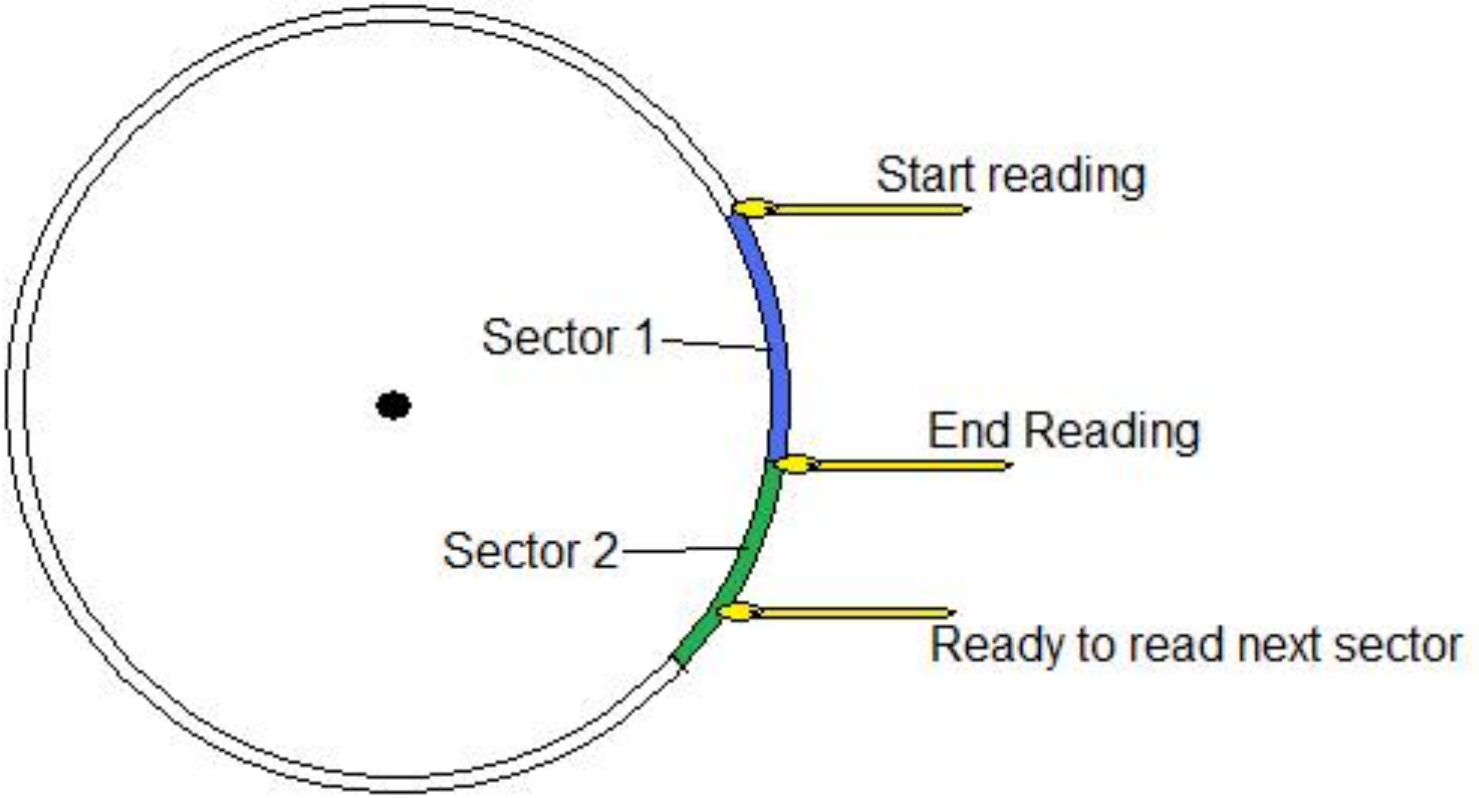
Inter-Sector Gaps

- Gaps include information like: sector start marker, Sector number, track number, etc



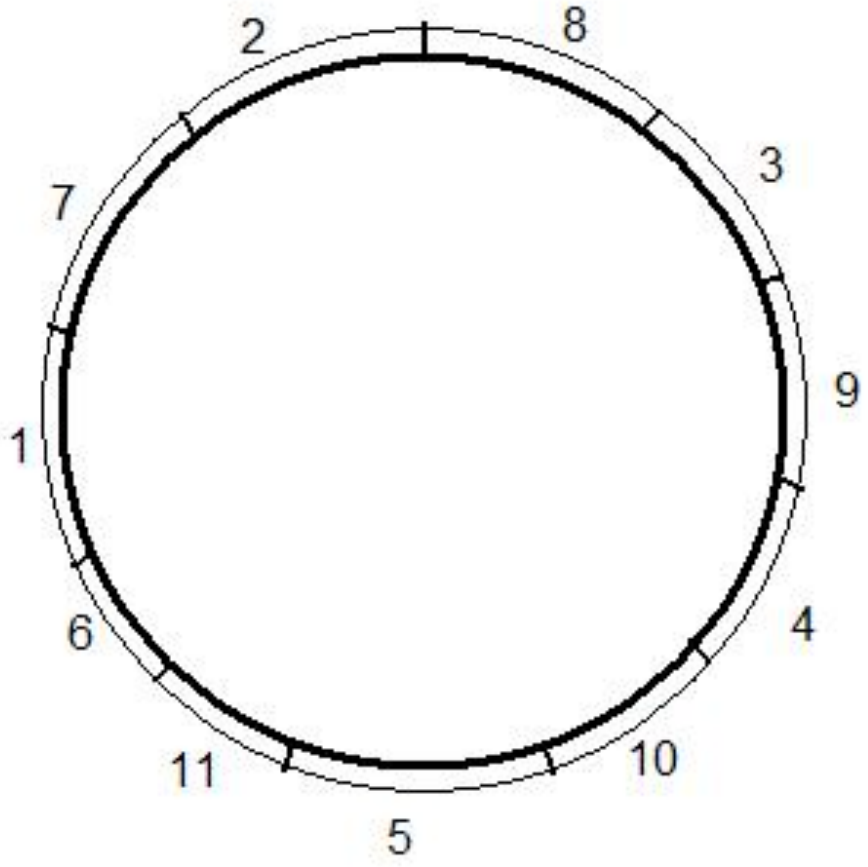
Interleaving

- Error checking the data in the sector will slow down the I/O operation



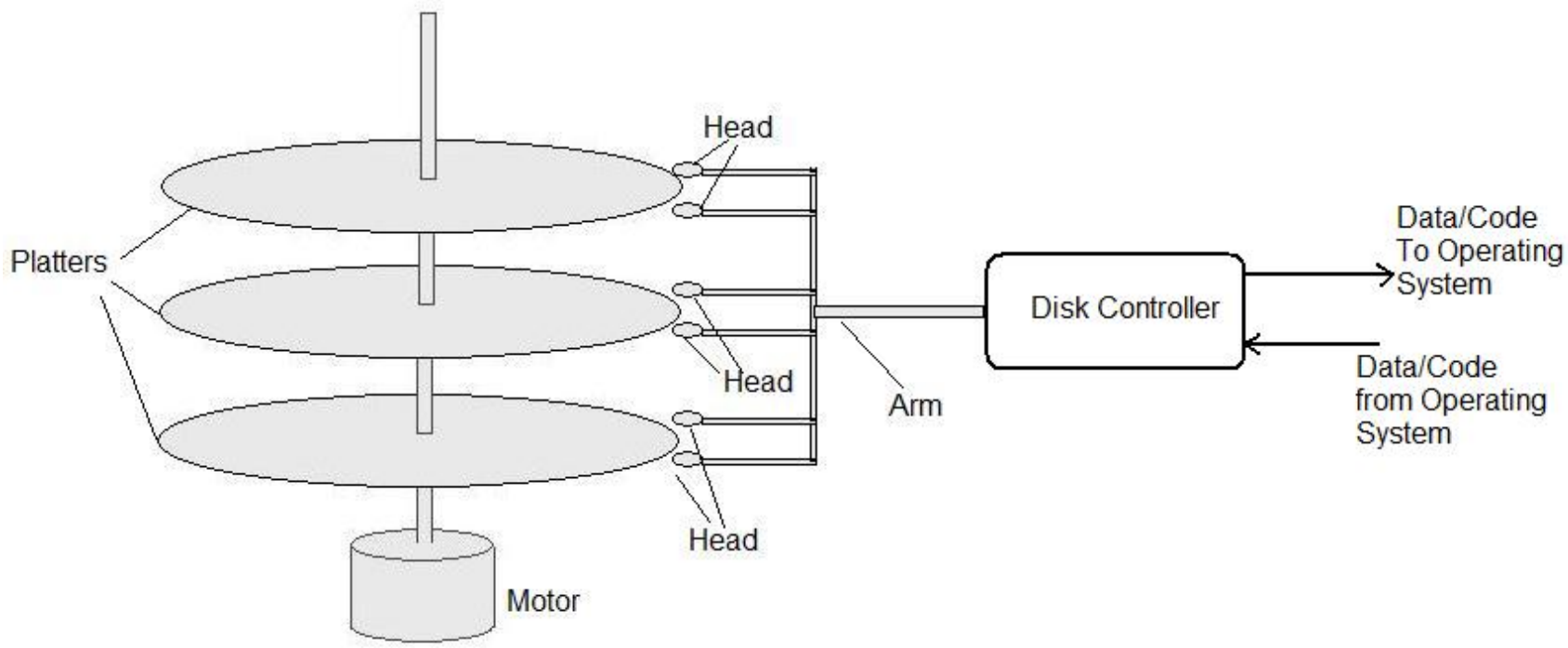
Interleaving

- Changing the order of numbering the sectors can speed up file I/O



Disk Controller

- Disk controller gets data/command from OS, controls I/O operation, and sends back the results to the OS



General Specifications of Hard Disks

- Hard drives spin a platter into the correct position for read/write.
- The spin has the rating of 5400 or 7200 rounds per minute (RPM)
- Hard drives can read and write data in the range of a few hundred megabytes (MB) per second.
- But we generally see much slower access.

- **Question:**
  - How can we speed up access to data?

Disk I/O Steps

- To read or write:
  - Move the head to the track
  - Find the sector
  - Transfer data to/from disk controller
  - Error checking and reporting to the OS

Disk I/O Timings

- ***Seek Time (S):*** The time needed for the head to move onto the track
- ***Rotational Latency Time (r):*** The time needed for the disk to rotate until the sector comes under the head
- ***Block Transfer Time (btt):*** The time needed to transfer data from head to sector (write) or sector to head (read)

Disk I/O Timing

Time to read/write a block : s+r+btt

Note: if the time needed for the head to pass over the gap is also considered then we have:

Time to read a block : s+r+ebt

ebt : Effective Block Transfer Time

Optimizing File I/O

- In many data processing applications, the data file is read from the beginning to the end.
- For these applications if the data is stored on
  - the same track
  - Or the neighboring tracks

The seek time (s) will be smaller, and the file I/O will be faster.

Blocks and Records

- A **Block** is the unit of I/O from a hard disk
- It is not possible to read a fraction of a block from a hard disk
- A **record** is the unit of information stored in a file. Example: Student Record (St. ID, St. Name, St. major, St. address, . . .)
- A **File** is a set of related records. Example: Hospital data file

Example 1

- Compute the time needed to read 10 consecutive blocks from the same track. Assume no interleaving.
- Use:

  s=16 msec

  r=8.3msec

  btt=0.8 msec

  ebt=0.84 msec

Solution

- Total time = s + r + 10 ebt

  = 16 + 8.3 + 10*0.84

  = 32.7 msec

Question: Why did we use ebt instead of btt?

Example 2

- Find the time needed to read 10 random blocks from the disk

  Use parameters from example 1

Solution

- Total time = 10 (s + r + btt)

  10 (16 + 8.3 + 0.8)

  251 msec

Question: Why btt is used here?

Buckets

- If the record and the block have different sizes, then several records are stored in a block.
- A Bucket is a group of records stored in a block
- The file read/write unit is bucket (not record!!)
- The data read from a file is put in a temporary place called a ***Buffer***

Blocking factor

- Blocking factor (Bfr) is the number of records in a block
- Example:

  Block size (B) = 2400 Bytes

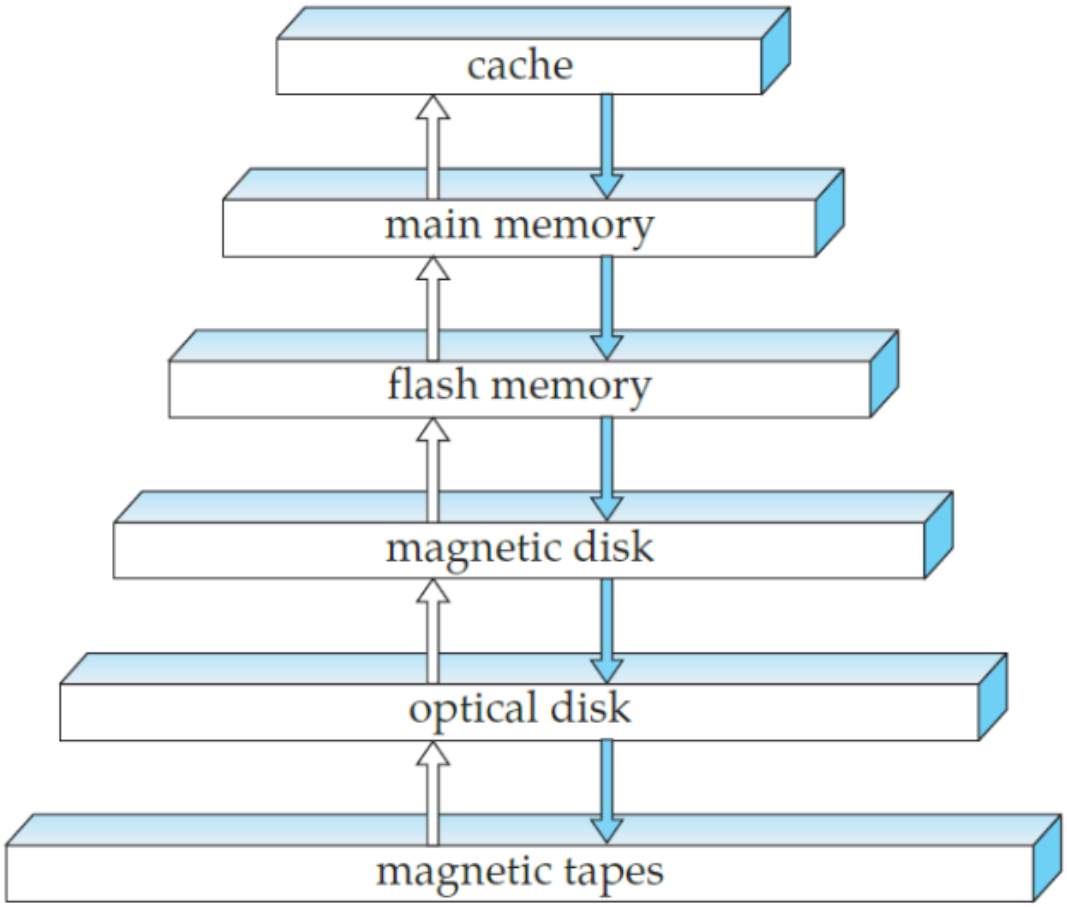
  Record size (R) = 100 Bytes

  Bfr = B/R = 2400 / 100 = 24

Double Buffering

- Buffer is a place to store blocks for processing
- When a block is processed, the disk reads the second block and puts it in a second buffer
- The role of the first and the second buffer is changed for the third block
- This use of two buffers is called double buffering

Other Types of Memory



EEPROMs

- Electrically Erasable Programmable Read Only Memory or EEPROMs are non-volatile memories.
- Instead of reading and writing information magnetically, EEPROM stores bits using semiconductor technology.
- EEPROM technology is used in flash memory, and SD cards.
- Data transfer rate of EEPROMs can be up to a few gigabytes

Solid State Drives

- Solid State Drives use EEPROM technology to store data
- SSDs have large capacity and can replace hard disks
- Benefits of SSDs
  - They do not have any spinning part so every block can be accessed randomly in less than a millisecond.
  - Their data access rate is much higher than hard disks
  - They consume less energy

Limitations of SSDs

- Each data unit can be used a limited number of times
- The number of times that a data unit can be used is between 10,000 up to a few million times.
- Therefore, the data should be distributed all over the SSD disks.

Do We Still Need File Access Optimizations?

- Data access in SSD drives does not include
  - Seek time (s)
  - Rotational latency time (r)
- Only Btt affects the total data transfer time

- Question:
  - Is file organization still necessary?

Example

- Assume we have a file with 100,000 blocks. Each block has the details of an employee.
- Case 1:
  - The blocks are not sorted (no order)
  - Find an employee data given his Employee ID
- Case 2:
  - The blocks are sorted with respect to the Employee ID
  - Find an employee data given his Employee ID

Case 1: Sorted File

- To find the employee record we use a binary search
- Considering the worst case, we will have to read 17 blocks
- Therefore the search time will be:

$$\text{Worst case search time} = 17 \text{ btt}$$

Case 2: Unsorted File

- To find the record we have to exhaustively read the blocks
- On average we will have to read half of the blocks. In the worst case we have to read all blocks

$$\text{Average search time} = 50{,}000 \text{ btt}$$

$$\text{Worst case search time} = 100{,}000 \text{ btt}$$

Summary

- In many real-life applications we have to deal with large amount of data.
- This data should be stored on secondary storage, such as hard disks
- Secondary storage memories are slow compared to the main memory.
- To speed up I/O operations, we should organize data

Questions?