Data Storage

Chapter 2

Introduction

- DBMS deal with large amounts of data
  - We cover the following issue in this chapter
    - How does computer systems store and manage large amounts of data ?

Outline

- Introduction
- The memory hierarchy
  - Cache
  - Main memory
  - Virtual memory
  - Moor’s law
  - Secondary Storage
  - Tertiary storage
  - Volatile and nonvolatile storage
- Disks
  - Mechanics of disks
  - The disk controller
  - Disk storage characteristics
  - Disk access characteristics
  - Writing blocks
  - Modifying blocks
- Using secondary storage efficiently
  - The I/O model of computation
  - Storing data in secondary storage
  - Merge sort
  - Two phase, multiway merge sort
  - Extension of multiway to larger relations

Outline

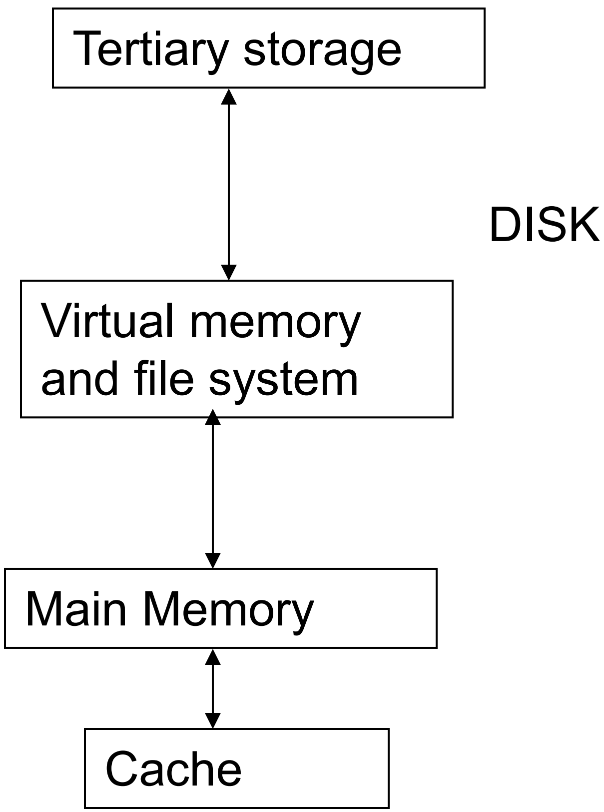
- Improving the access time of secondary storage
  - Organizing data by cylinders
  - Using multiple disks
  - Mirroring disks
  - Disk scheduling and elevator algorithm
  - Prefetching and large-scale buffering
  - Summary of strategies and tradeoffs
- Disk failures
  - Intermittent failures
  - Checksums
  - Stable storage
  - Error handling capabilities of stable storage
  - Recovery from disk crashes
  - The failure model of disks
  - Mirroring as a redundancy tencnique
  - Parity blocks
  - RAID 5
  - Coping with multiple disk crashes

The memory hierarchy

- Processor
  - Fast, reduced instruction set, with cache, pipelined…

  Speed: 100 $\rightarrow$ 500 $\rightarrow$ 1000 MIPS
- Speed 10 nanoseconds

(10^{-8} seconds) or less



Cache

- Copy of instructions in main memory
- Unit of transfer is small number of bytes
  - On board cache
  - L2 cache
- In a single computer, no need to update the main memory when a cache is modified
- In case of muiltiprocessor system
  - Update to cache should be reflected in main memory
- Speed between cache and main memory 100 nano seconds (10^{-7} seconds)

Main memory

- Important component
  - 100 MB and GB range are available
- Random access
  - One can obtain any byte in the same time
- If the data is in main memory the access time is 10 to 100 nsec range.(10^{-8} to 10^{-7} sec range.

Virtual Memory

- The program/data ocupies virtual memory address space.
- Many machines have 32 bit address space
  - Can address 2^{32} or about 4 billion different addresses.
  - So typical virtual memory is 4 GB.
- Since virtual memory is much bigger than main memory most of the content is stored in disk.
- Large scale database systems mange the data on disk.

Moore’s law

- Integrated circuits are improving and performance doubles every 18 months
  - Speed of processors
  - Cost of main memory per bit
  - Cost of disk per unit
- The following parameters do not follow Moore’s law
  - Speed of accessing data in main memory
  - Speed of disk rotation
- Processor computation time is reducing
- Relatively the distance of disk from CPU is increasing.

Secondary Storage

- Disk is used to store virtual memory and file system
- When the file is open for reading, the OS reserves one block of man memory as a buffer for a file. After consuming the data in the block, another block is trandferred.
- DBMS manages the blocks itself.
- It takes 10 to 30 msec to read or write the block on the disk

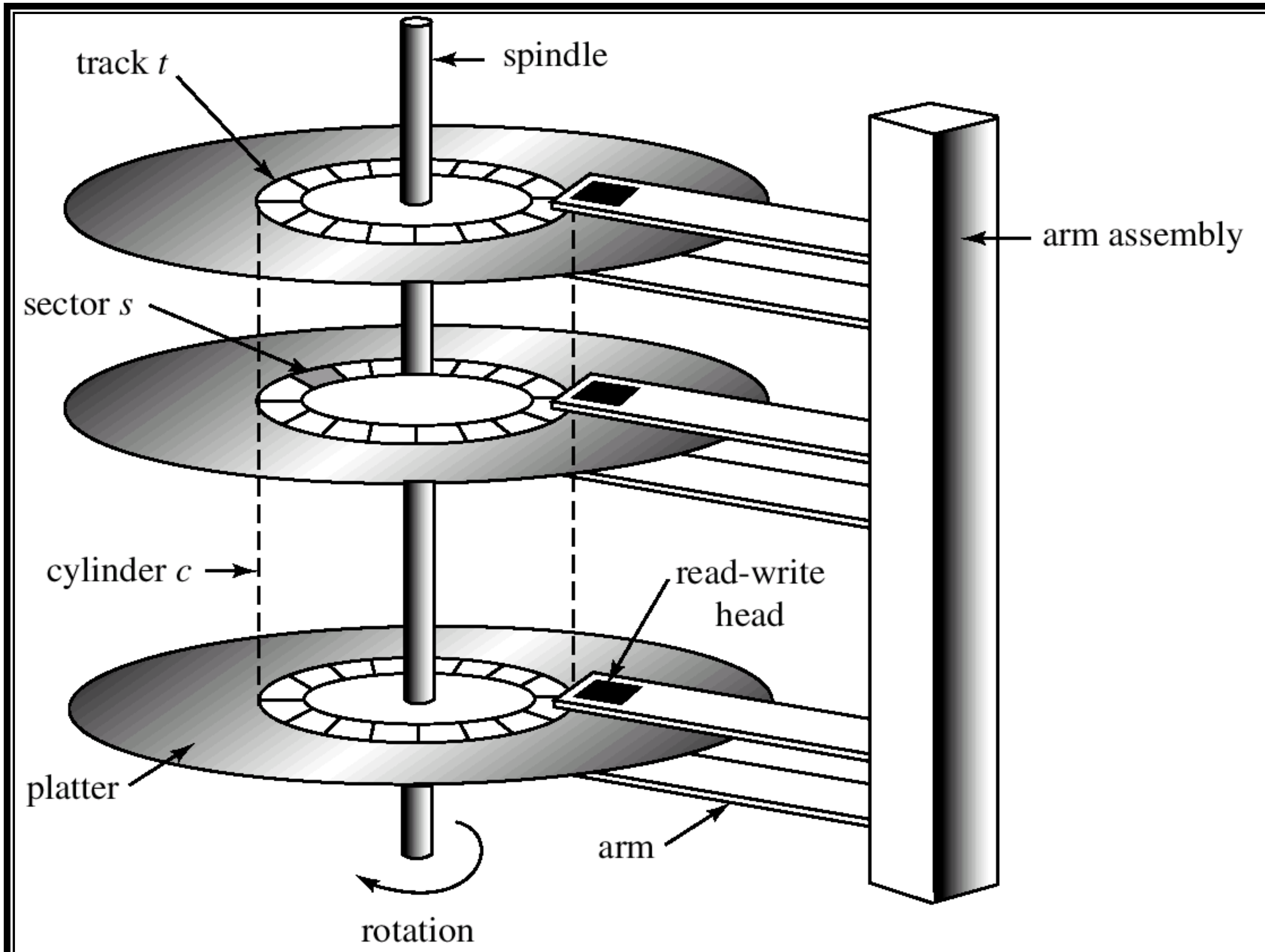
Tertiary Storage

- There are databases bigger than disks
- Teriary storage devices have been developed to hold data volumes measured in tera bytes.
- Tertiary storage access times 1000 times slower than disk access times.

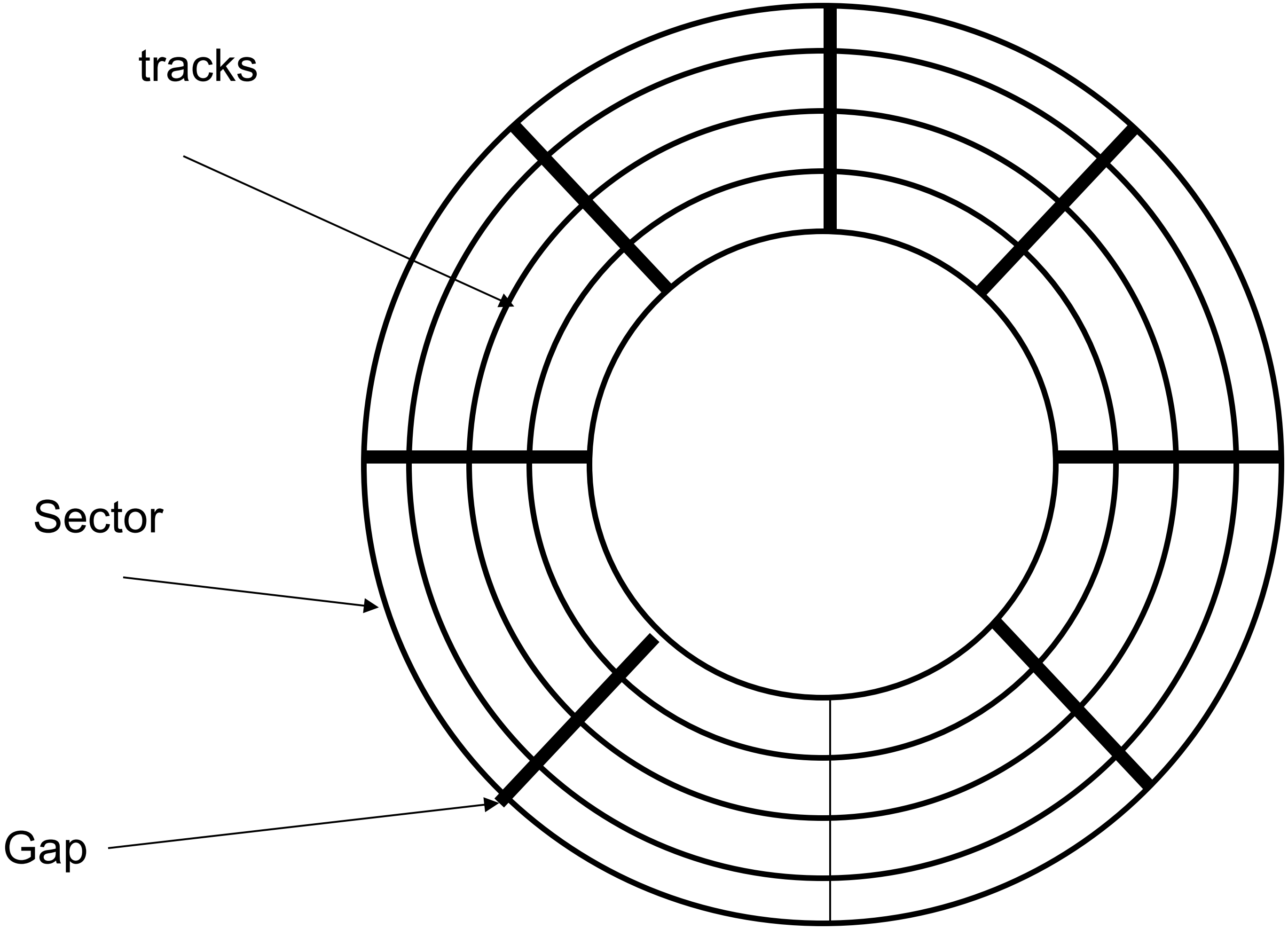
Volatile and Nonvolatile Storage

- Storage can be volatile or nonvolatile
  - Volatile
    - Forgets the data when the power goes-off
  - Nonvolatile
    - Contents are intact when the power goes-off.

Disks
Mechanics of Disks



Top View



Mechanics of Disks

- Platter, Head, Actuator, Cylinder, Track, Sector (physical), Block (logical), Gap
- Disk assembly
  - Consists of one or more disk platters that rotate around a central spindle
- Tracks
  - Data are stored into tracks which are concentric circles on a single platter
  - Tracks represent many points each represent a single bit by direction of magnetism
- Sectors
  - Tracks are organized into sectors
  - The circle is separates by gaps
  - It is an indivisible unit as far as reading or writing of disk is concerned.
  - If the sector is corrupted, it can not be used.
- Gaps represent 10 % of total area and help to identify the beginning of sectors.
- Blocks of logical units of data consists of one or more sectors.

Mechanics of Disks

- Head assembly holds disk heads
    - One head for each surface
- The disk controller capable of
    - – Controls the mechanical actuator that moves head assembly. The tracks under the heads at the same time are said to form a cylinder
    - – Selecting the surface and a secor for read and write
    - – Transferring the bits read from the desired sector to the main memory and writing the main memory contents to the disk.

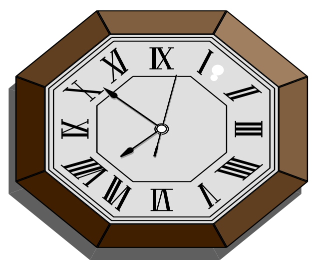
Disk Storage Characteristics

- Measures associated with the disk
  - Rotation speed of disk assembly
    - 5400 RPM, one rotation for every 11 msec.
  - Number of platters per unit
    - Typical disk has four to five platters
  - Number of tracks per surface
    - A surface may have 10,000 tracks
  - Number of buts per track
    - 10^5 or more bytes per track.

Data Access Characteristics

Disk Access Time

I want
block X



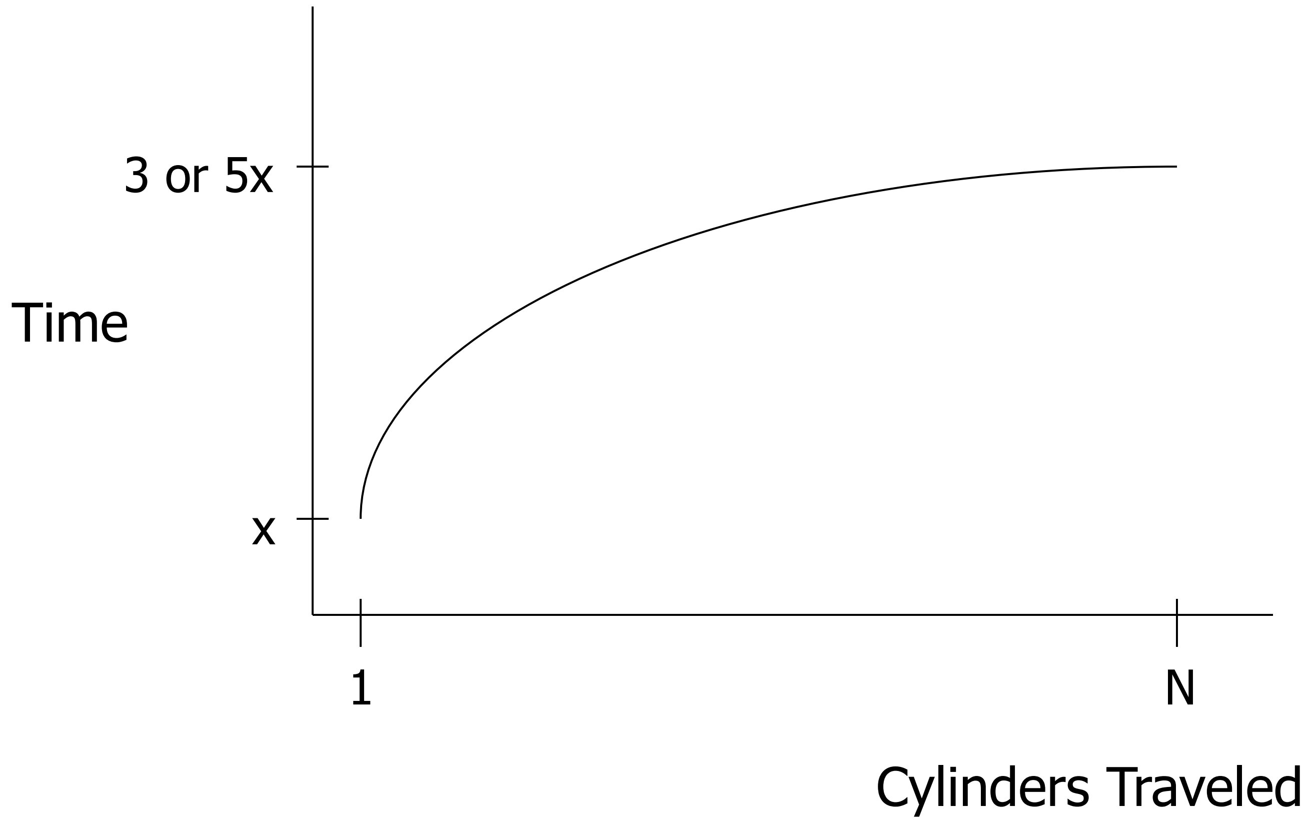
?

block x
in memory

Disk Access Time

- Disk access time can be broken into two parts
  - Processing time
  - Seek time: moving head assembly at the proper cylinder
  - Rotational latency: time for the disk to rotate to the first sector
  - Transfer time:
    - Less than msec
- Time = Seek Time +
  Rotational Delay +
  Transfer Time +
  Other

Seek Time



Average Random Seek Time

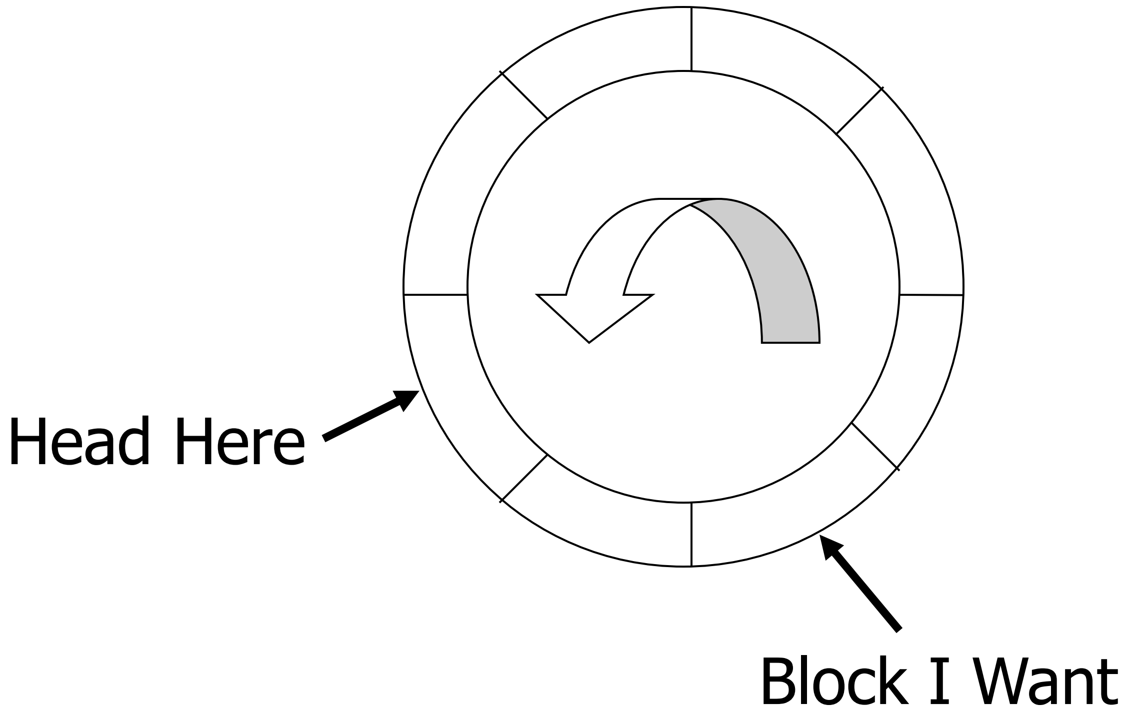
$$S = \frac{\displaystyle\sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} \text{SEEKTIME } (i \rightarrow j)}{N(N-1)}$$

Average Random Seek Time

$$S = \frac{\displaystyle\sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} \text{SEEKTIME } (i \rightarrow j)}{N(N-1)}$$

"Typical" S: 10 ms $\rightarrow$ 40 ms

Rotational Delay



Average Rotational Delay

R = 1/2 revolution

"typical" R = 8.33 ms (3600 RPM)

<u>Transfer Rate:</u> t

- "typical" t: 1 $\rightarrow$ 3 MB/second
- transfer time: $\dfrac{\text{block size}}{t}$

Other Delays

- CPU time to issue I/O
- Contention for controller
- Contention for bus, memory

Other Delays

- CPU time to issue I/O
- Contention for controller
- Contention for bus, memory

"Typical" Value: 0

- So far: Random Block Access
- What about: Reading “Next” block?

If we do things right (e.g., Double Buffer, Stagger Blocks…)

$$\text{Time to get block} = \frac{\text{Block Size}}{t} + \text{Negligible}$$

- skip gap
- switch track
- once in a while, next cylinder

Rule of Thumb	Random I/O: Expensive Sequential I/O: Much less

- Ex: 1 KB Block
  - » Random I/O: $\sim$ 20 ms.
  - » Sequential I/O: $\sim$ 1 ms.

Cost for Writing similar to Reading

…. unless we want to verify!
need to add (full) rotation + $\frac{\text{Block size}}{t}$

- To <u>Modify</u> a Block?

- To Modify a Block?

To Modify Block:

(a) Read Block

(b) Modify in Memory

(c) Write Block

[(d) Verify?]

Block Address:

- Physical Device
- Cylinder #
- Surface #
- Sector

Using Secondary Storage Effectively

- RAM model of computation
  - In most algorithms it is assumed that data is in main memory
- We have to design algorithms that limit the disk accesses
  - Logic can be extended to any level
- I/O model of computation
  - Data is in disk. I/O cost dominates the processing time.

Sorting the data in secondary storage

- If the data fits in main memory quick sort can be used.
  - Classic merge-sort algorithm
    - Divide the list arbitrarily into two lists
    - Recursively sort the two lists.
    - Merge the sorted lists.
- Two phase, Multiway merge sort
  - Sort main-memory sized pieces of data
  - Merge the all the sorted lists into a single sorted list.

Merging the sorted lists

- Merging by pairs results into $2\log_2 n$ times reading of data.
- The following algorithm is followed
  – Find the smallest key among the first remaining elements of the all the lists
    - Linear search
  – Move the smallest element to output list
  – If the output block is full, re-initialize same buffer for the next output block
  – If the input block is empty, get the next elements of the same sorted list, if there are no elements, leave it empty.

Extension of Multiway merging to Larger Relations

- The two-phase multiway Merger-sort can be used to sort large number of records.
  - Let the block size is B bytes
  - Main memory available for buffering blocks is M bytes
  - Record takes R bytes.
  - The number of buffers available= M/B.
    - One is for output buffer
  - The number of sorted sub-lists= (M/B)-1
    - Number of times we fill main memory with records to be sorted.
  - Each time we fill the memory we sort M/R records.
  - The total number of records we can sort is (M/R)((M/B)-1)= M^2/RB records.
- Suppose M=50,000,000, B=4096 and R=100 we can sort up to M^2/RB=6.1 billion records, occupying six tenth of tera byte.
- If we want to sort more records, we can add a third phase.
  - What is the size ?

Improving the access time of secondary storage

- If the blocks are placed randomly, it is good to execute small queries.
- If the system is sorting a large relation, it is better to store the data in consecutive blocks.
- Some strategies
  - Place the blocks that are accessed together in the same cylinder
  - Divide the data into several small disks rather than one.
  - Mirror the disk
    - Reliability and parallelism
  - Use the disk scheduling algorithm
    - OS, disk scheduler or DBMS should select the order in which several request bocks will be read or written.
  - Pre-fetch the blocks to main memory in anticipation of their later use.
- We discuss the following
  - Dedicated case
  - Multiple processes and mix of queries case.

Organizing data in cylinders

- Seek time represents about half the average time to access the block.
- If we choose to read all the blocks on a single track or cylinder, we can ignore the first seek time.
- So, store the blocks in one cylinder
  – If exceeds, store in adjacent cylinders.

Using Multiple disks

- Replace one disk with multiple disks, with heads locked together
- Resulting effect
  - All (reading, writing) times are divided by number of disks.

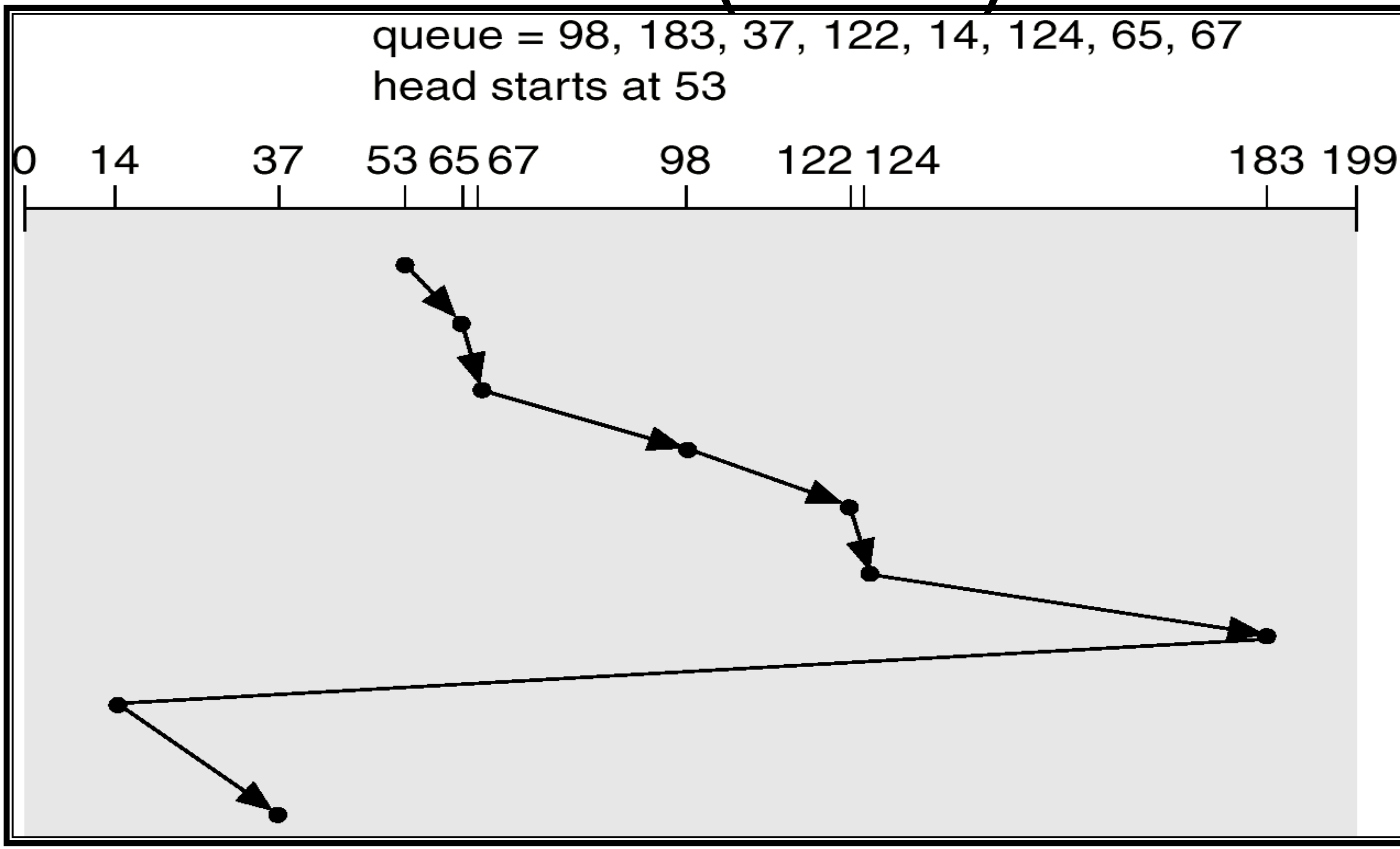
Mirroring disks

- Make disks to have identical copies of data
- They do not speed-up writing.

Disk scheduling and Elevator algorithm

- Elevator algorithm
  - Disk head makes sweeps across the disk, from innermost to outermost cylinder them back again.
  - Like an elevator
- If the average number of waiting for the disks increases, the elevator algorithm further improves the throughput.

C-LOOK (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

Pre-fetching and Large-scale buffering

- Pre-fetching or double buffering
  - In applications where we predict the order of blocks.
- Fetch all the data
  - Load all the buffers before they are needed.
- Merge sort
  - Read whole tracks or whole cylinders whenever we need some more records from a given list.
- Requires extra memory buffers

Disk Failures

- Types of failures
  - Intermittent failures
    - Attempt to read/write a sector is unsuccessful, but repeated tries are able to read
  - Media decay:
    - Bits or bits are permanently corrupted.
  - Write failure
    - Unable to write, or unable to read the written data
  - Disk crash
    - Entire disk becomes unreadable.

Intermittent failures

- Disk sectors store redundant bits
  - Purpose of redundant bits
    - To know whether written data is correct or not
- A reading function returns a pair (w,s), where “w” is a data in the sector that is read and “s” is the status bit that tells whether the data read is correct or not.
- In the intermittent failure the status is bad several times.
- Checksums are used
  - parity

Stable Storage

- Checksum does not help in correcting the error
  - What will happen if write the data and unable to read the data ?
  - If we overwrite the data and unable to read the old and new data ?
- Implement the stable storage on several sectors or several disks
- Stable storage Idea
  - Each sector is paired. Each pair represents one sector
    - Left and right copies (XL and XR)
  - Stable write
    - (1)Write the value of X into XL.
      - Repeat the write as long as status is returned as good
      - Otherwise, select another sector in place of XL
    - Repeat (1) for XR
  - Stable read
    - (2) if the status is bad, repeat the read the number of times.
    - If we can not read XL,repeat (2) with XR.

Error handling capabilities of stable storage

- Media failures
  - If one of the sector fails, we can always read from the other
- Write failures
  - If the power fails while writing, one of the sector goes bad.
    - We are able to determine the old value of sector.
  - If the failure occurs after writing XL, new value can be read.

Recovery from Disk Crashes (Head Crash)

- Failure model of disks
- Mean time to failure.
  - It is a length of time by which 50% of a population of disks have failed catastrophically.
  - about 10 years
- Assuming linear assumption
  - If 50 percent failed in 10 years, then 5 % fail in first year.
- Disk crash does not mean data loss
  - Alternative schemes are used for data.

Recovery from Disk crashes

- Disk Mirroring
  - One is data disk and another is mirror or redundant disk
  - Referred as RAID-level 1
  - Mean time to memory loss is very higher than mean time to disk failure
  - If there is a disk crash which first disk is repaired or copied, there is a problem

Disk failure

- Suppose, replacing a failed disk takes 3 hours (1/8 of a day or 1/2920 of a year).
- Disk failure rate is 5%
- The probability that mirror disk fails during copying is
  - (1/20)*(1/2920) or one in 58, 400.
- If the disk fails one in 10 years, one of the two disks will fail once in 5 years.
- Mean time to failure is: 5*58,400=292,000 years.

RAID Structure

- **Problem with disk**
  - **Data transfer rate is limited by serial access.**
  - **Reliability**
- **Solution to both problems: Redundant arrays of inexpensive disks (RAID)**
- In the past RAID (combination of cheap disks) is alternative for large and expensive disks.
- Today: RAID is used for their higher reliability and higher data-transfer rate.
- So the I in RAID stands for “independent” instead of ‘inexpensive”.
  - So RAID stands for **Redundant Arrays of Independent Disks.**

- RAID is arranged into six different levels.

RAID: Improvement of Reliability via Redundancy

- The chance that some disk out of N disk fail is much higher than single disk.
- Each failure of disk leads to loss of data
- Solution: Redundancy.
  – Store extra information which can be used in the event of failure.
- Simplest: Mirrored disks
  – Each logical disk consists of two physical disks.
  – Read from any disk
  – Write to both disks.

RAID: improvement in Performance via Parallelism.

- # of reads per unit time can be increased.
- With multiple disks we can improve the transfer rate with data stripping.
- Bit level Data stripping:
  - Splitting the bits of each byte across multiple disks.
  - If we have array of 8 disks, we write bit i of every byte to disk i.
  - The array of eight disks can be treated as single disk that are eight time normal size and eight times the access rate.
  - Every disk participates in the read.
  - But each access can read eight times as many data.
- Block level stripping: Blocks of files are stripped across multiple disks.
- Two goals
  - Increase the throughput of multiple small accesses.
  - Reduce the response time of large accesses.

Block-level stripping

- Block-level stripping is the logical extension of bit-level stripping.
- For example first block contains 1000 bytes. So it is a sequence of 7000 bits (ignoring 8th bit which is a parity bit). The first block is stored n Disk1. Similarly, another blocks are stored in Disk2, Disk3,..., Disk7 respectively. In Disk8, the parity of D1 to D7 blocks are stored.
- Disk1: 7000 bits of block1
- Disk2: 7000 bits of block2
- Disk3: 7000 bits of block3
- .
- .
- Disk7= 7000 bits of block7
- D8= 7000 bit parity (parity of 7 bits (combination of first bit of each block)).

- So if any of the disk fails, we can easily recover the failed data by accessing other 7 disks.

RAID levels

- RAID level 0: Disk arrays with stripping at the level of blocks, but without any redundancy (no mirroring and no parity)
  - Block level stripping
- RAID level1: Disk mirroring.
  - Block level striping
- RAID level2:
  - Error detection with parity bits.
  - Error correcting stores two or more parity bits.
  - Data can be stripped among disks and parity bits are stored in other disks.
  - Bit level stripping

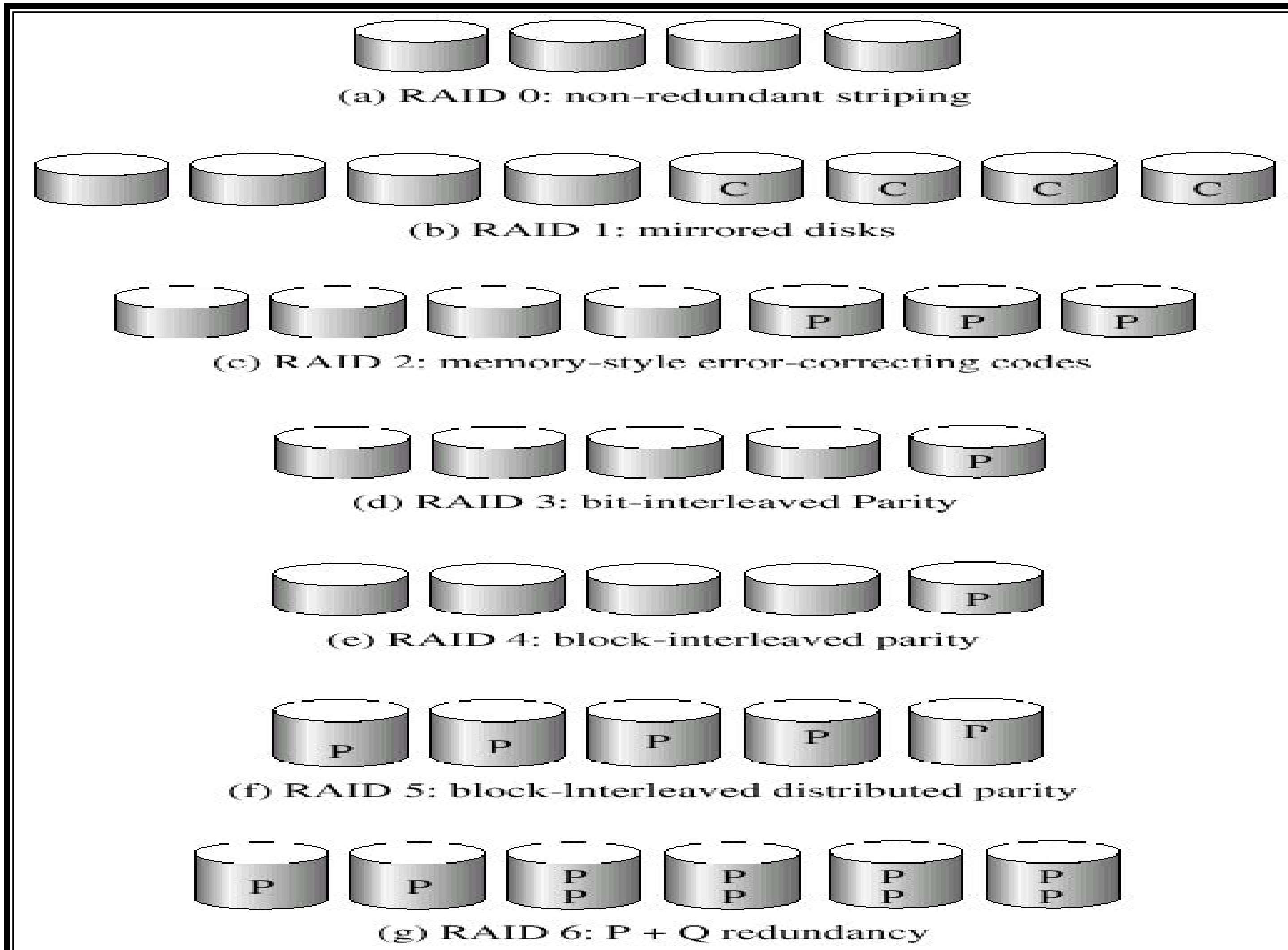
RAID levels

- RAID level 3: bit-interleaved parity organization
  - Disk controllers can detect whether a sector has been read correctly.
  - The bits of failed sector can be recovered by computing the parity of the remaining bits.
- RAID Level 3 is similar to RAID level 2 but less expensive
  - One disk overhead.
- RAID level 2 is not used in practice.
- Advantages of RAID level 3 over level 1 (mirroring)
  - One parity disk is needed; reducing the storage overhead
  - Transfer rate is same.
- RAID 3 performance problem
  - Computing and writing parity

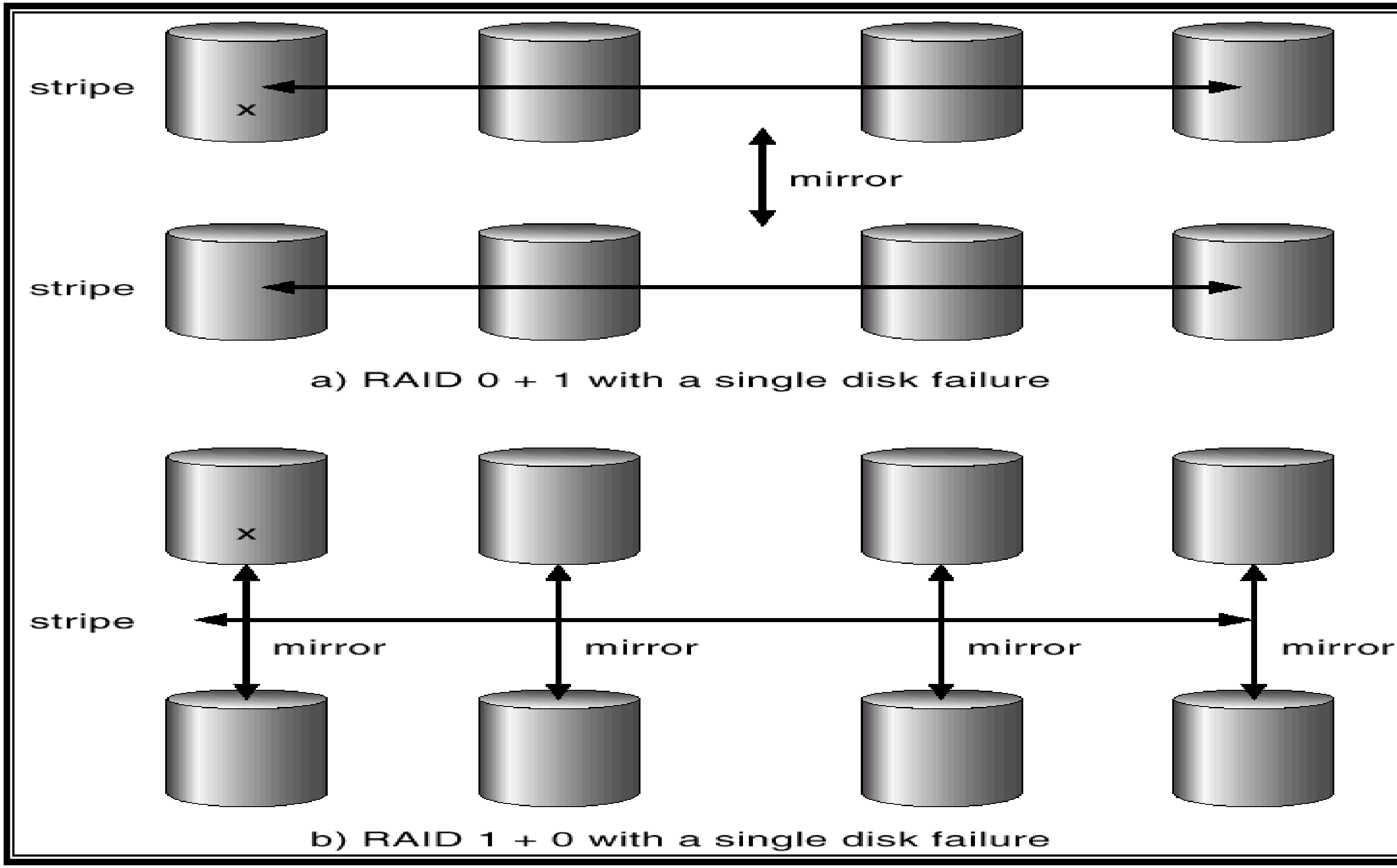
RAID levels

- RAID level 4: block-interleaved parity organization
  - Uses block-level stripping
  - Keeps parity block on separate disk
  - If one disk fails the parity block and other blocks can be used to recover the failed disk.
- A block read accesses only one disk
  - Data transfer rate for each process is slower, However multiple read requests can be carried out in parallel.
  - Write results into two writes: block and corresponding parity.
- RAID level 5: Block interleaved distributed parity
  - Parity is distributed among all N+1 disks.
  - For each block one disk stores parity and others store data.
- RAID level 6:
  - Similar to RAID 5, but stores extra redundant information to guard against multiple disk failures.
- RAID 0+1 and RAID 1+0
  - Combination of RAID levels 0 and 1

RAID Levels


C
C
C
C
(a) RAID 0: non-redundant striping
(b) RAID 1: mirrored disks
P
P
P
(c) RAID 2: memory-style error-correcting codes
P
(d) RAID 3: bit-interleaved Parity
P
(e) RAID 4: block-interleaved parity
P
P
P
P
P
(f) RAID 5: block-lnterleaved distributed parity
P
P
P P
P P
P P
P P
(g) RAID 6: P + Q redundancy

RAID (0 + 1) and (1 + 0)



Selecting a RAID level

- In case of a failure, the time to rebuild data vary with the RAID level.
- RAID level 0: used for high performance applications where data loss is not critical.
- RAID level 1: Popular for applications that require high reliability with fast recovery.
- RAID 0+1 and 1+0 are used where performance and reliability are important.
- RAID level 5 is used to store large volume of data.
- RAID level 6 is not supported.
- Hot spare disks can be used to reduce human intervention.
- The concepts of RAID can be generalized to other systems
  - Arrays of tapes
    - Recovery of damaged tape
  - Broadcast of data over wireless systems.
    - Receiver can reconstruct the information with parity information.

Multiple disk crashes

- Use more redundant disks