



Data Storage

Chapter 11 of GUW



Objectives

- To understand how a computer system store and manage very large amounts of data.
- See how the efficiency of algorithms involving very large amount of data depends on the pattern of data movement between the primary, secondary, and tertiary storages.



- Chapter outline

- **The Megatron Database System** ←
- The Memory Hierarchy
- Disks
- Using Secondary Storage Effectively
- Accelerating Access to Secondary Storage
- Disk Failures
- Recovery from Disk Crashes
- Summary
- References
- Reading List



- The Megatron Database System

- What is Megatron?
- Megatron Implementation Detail
- How Megatron Executes Queries
- What is Wrong with Megatron?



-- What is Megatron?

- Magatron Systems Inc. is a fictitious software company
- Magatron is a Relational DBMS developed by Magatron Systems Inc
 - In this lecture and the following few lectures, we will use Magatron RDBMS to explain the implementation of RDMS.
 - Let us call the Megatron RDBMS as **MT**.



-- MT Implementation Detail ...

- Uses Unix file system to store its relations
- The relation Student(name, id, dept)
 - Is stored in /usr/db/Students
 - Has one line per tuple
 - Field values are stored as strings and separated by the character #
 - Example:

Smith#123#CS
Jhonson#522#EE

...

- The database schema is stored in /usr/db/schema
 - Example:

Students#name#STR#id#INT#dept#STR
Dept#name#STR#office#STR



... -- MT Implementation Detail

- Assume the Unix prompt is:

UX>

- To invoke MT run the command:

Megatron

- Magatron prompt is:

MT>

- To ends MT SQL query type the charcter:

#

- Example:

UX> megatron

MT> SELECT * FROM Students **#**

- Produces:

Name	Id	dept
Smith	123	CS
Jhonson	522	EE

MT> SELECT *

FROM Students | fileout.dat **#**

- Creates a new file /sur/db/fileout.dat



-- How MT Executes Queries ...

- Example 1:

- To process: `SELECT * FROM R WHERE <condition>` MT will do the following:
 1. Read the schema to determine the attributes of R
 2. Check the <condition> is semantically valid for R
 3. Display each of the attribute names as the header of a column, and draw a line.
 4. Read file name R, and for each line:
 - A. Check the condition, and
 - B. Display the line as a tuple, if the condition is true.



... -- How MT Executes Queries

- Example 2:

```
SELECT office
FROM Students, Depts
WHERE Students.name = 'Smith'
AND Students.dept = Depts.name #
```

- The algorithms used by MT02 can be:

```
For each tuple s in Students DO
  For each tuple d in Depts DO
    IF s and d satisfy the where-condition THEN
      display the office value from Depts
```



- What is Wrong With MT?

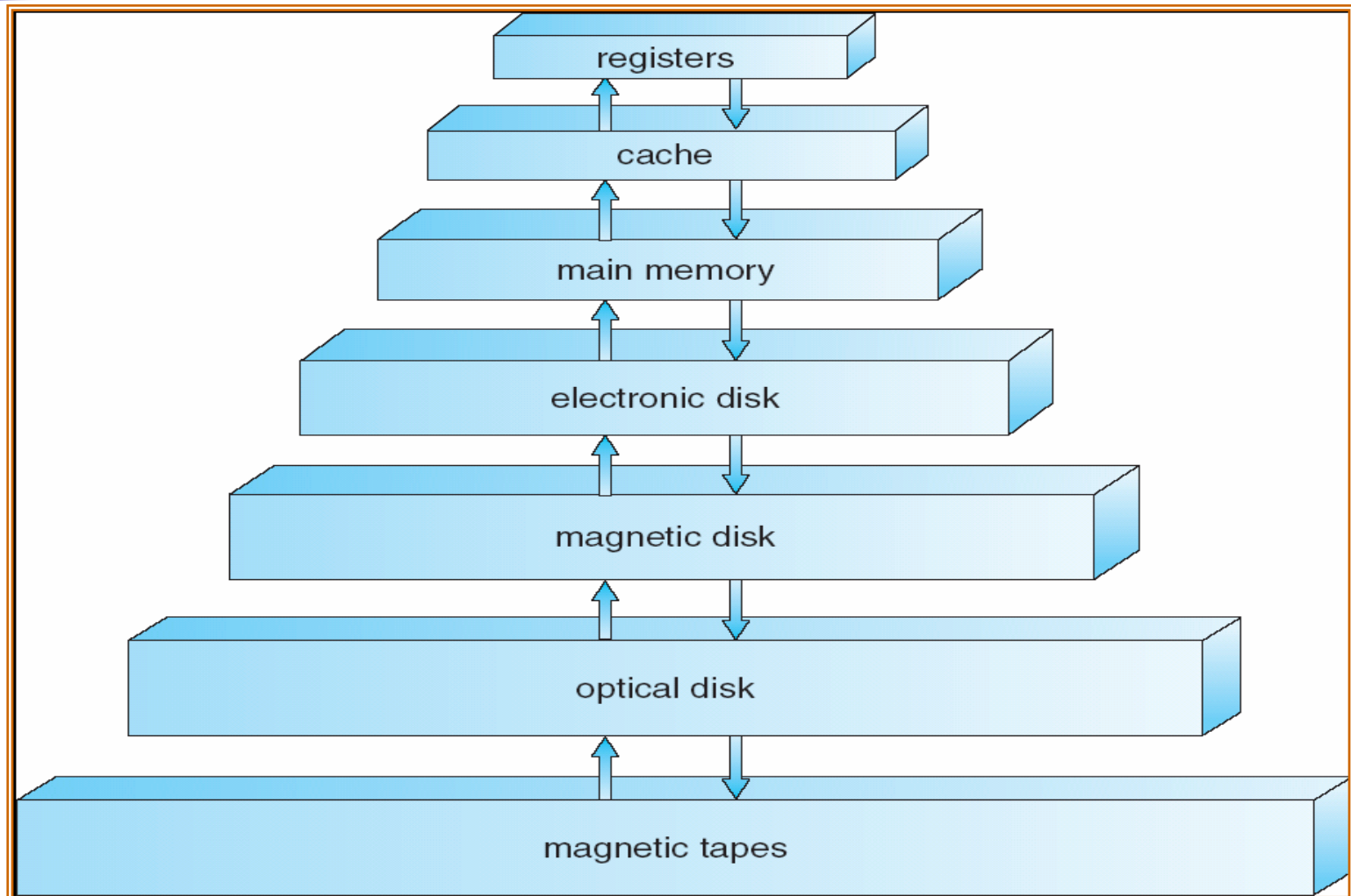
- DBMS is not really implemented like MT.
- M is inadequate for applications involving significant amount of data or multiple users.
- A partial list of problems are:
 - The tuple layout on disk is not flexible
 - Search is very expensive. We have to read the file sequentially.
 - Query processing, like the join operation, can be done in much better way.
 - Useful data is not buffered in the memory. They must always come from disk.
 - No concurrency control
 - There is no reliability. We can lose data for different reasons.



- Chapter outline

- The Megatron Database System
- **The Memory Hierarchy** ←
- Disks
- Using Secondary Storage Effectively
- Accelerating Access to Secondary Storage
- Disk Failures
- Recovery from Disk Crashes
- Summary
- References
- Reading List

- The memory Hierarchy





-- How far away is data?

Location	Cycles
Registers	1
On-chip cache	2
On-board cache	10
Memory	100
Disk	10^{**6}
Tape	10^{**9}

Location	Time
My head	1 min
This room	2 min
KFUPM Campus	10 min
Dhahran	1.5 hours
Pluto	2 years
Andromeda	2000 years

(Source: AlphaSort paper, 1995)



- Chapter outline

- The Megatron Database System
- **The Memory Hierarchy**
- **Disks** ←
- Using Secondary Storage Effectively
- Accelerating Access to Secondary Storage
- Disk Failures
- Recovery from Disk Crashes
- Summary
- References
- Reading List



- Disks

- Disks and DBMS
- Mechanics of Disks
- The Disk Controller
- Disk storage Characteristics
- Disk Access Characteristics
- Writing Blocks
- Block address
- Modifying Blocks
- Why Read and Write in blocks



-- Disks and DBMS

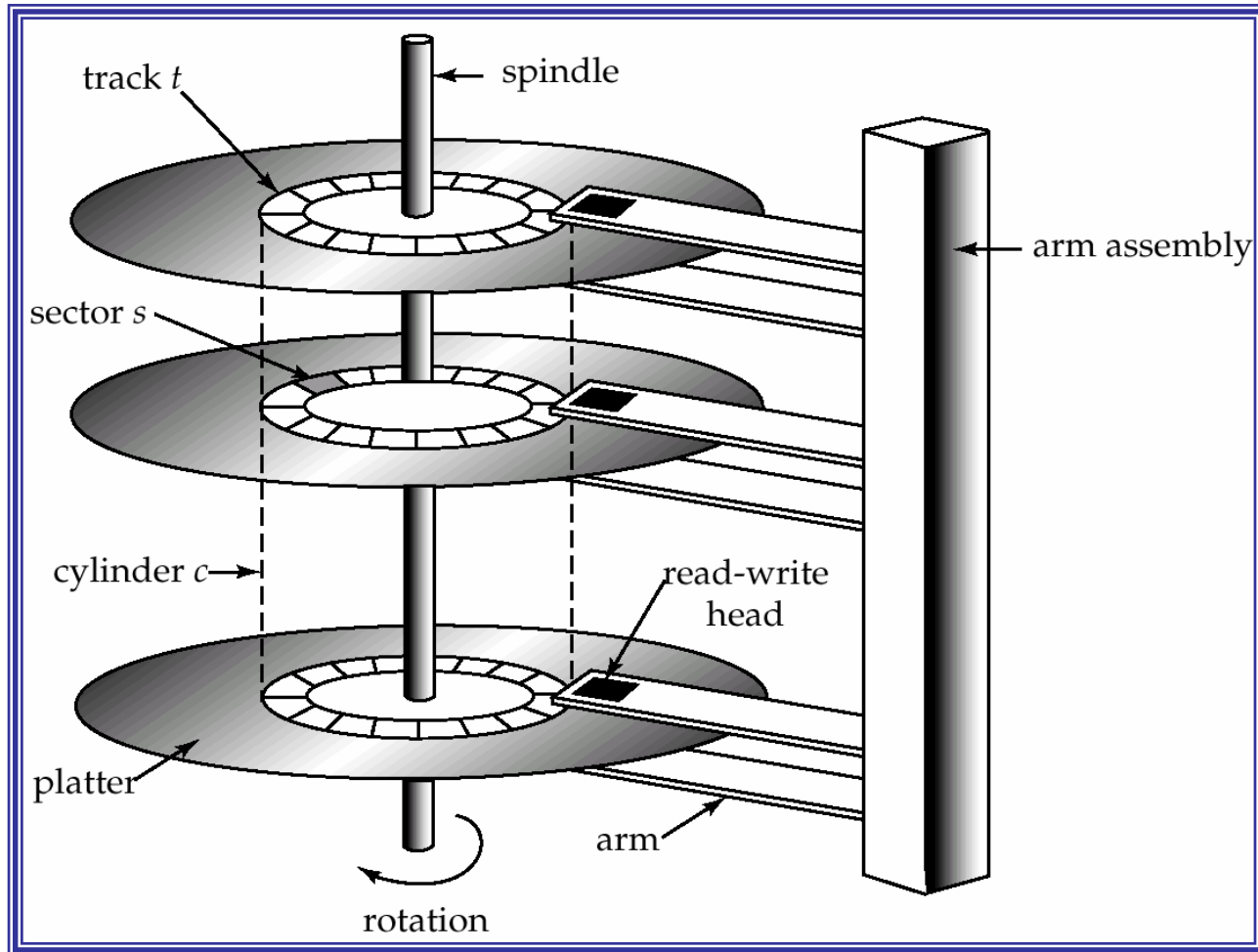
- A secondary storage is one of the important characteristics of DBMS.
- Most of the DBMS data and programs are stored in secondary storage.
- Secondary storage is almost exclusively based on magnetic disks.
- The way data is stored and retrieved from disks affects the performance of the DBMS.
- So the question is how to optimize the use of disks for a DBMS.



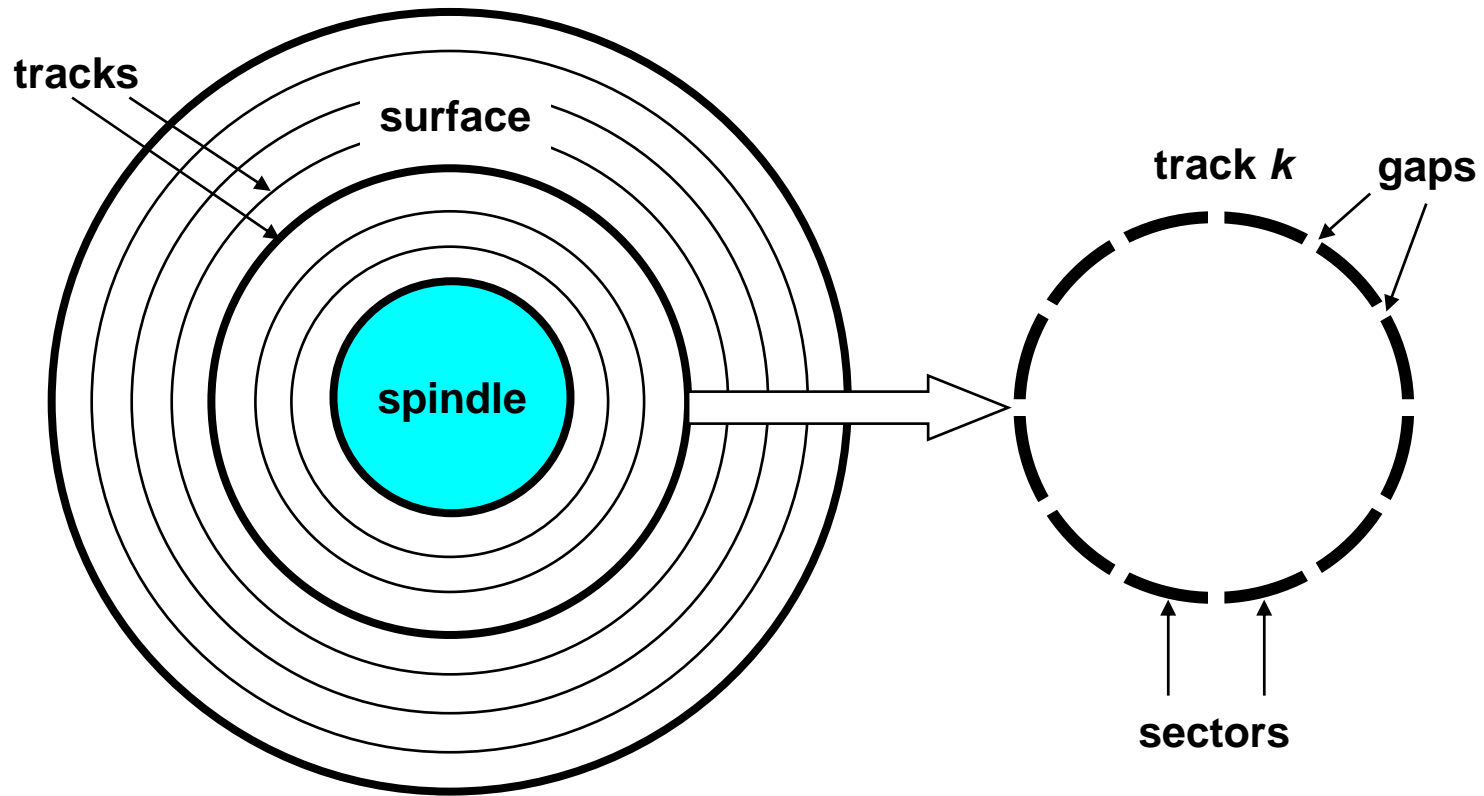
-- Mechanics of Disk ...

- Disks consist of **platters**, each with two **surfaces**.
- Each surface consists of concentric rings called **tracks**.
- Corresponding tracks from each platter form a cylinder.
- Each track consists of **sectors** separated by **gaps (10%)**. Sector is unadvisable unit as far as:
 - Reading
 - Writing
 - Errors
- **Blocks:**
 - are logical units of data that are transferred between disk and main memory
 - Consist of one or more sectors.

-- Mechanics of Disk ...



... -- Mechanic of Disk

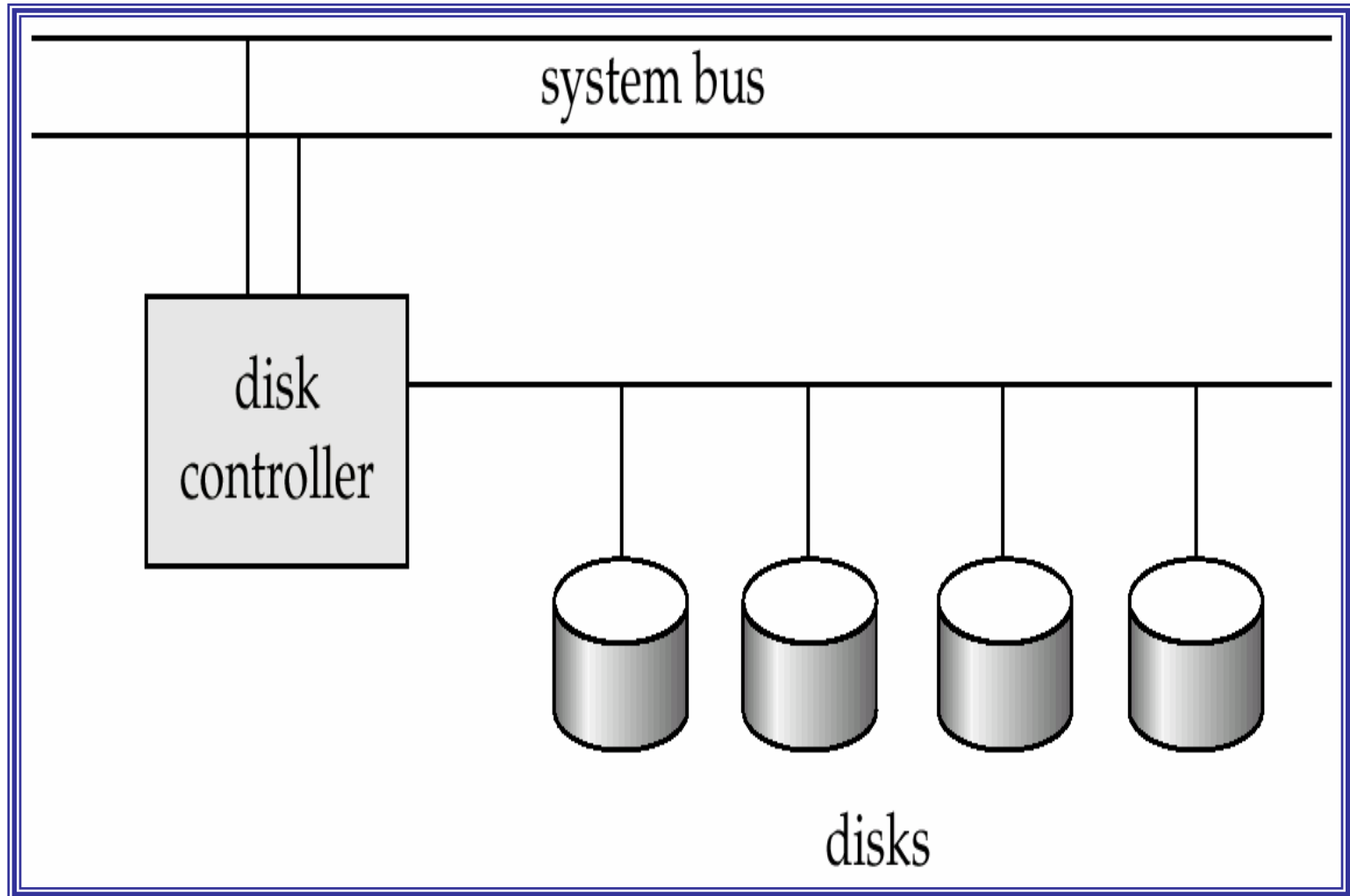




-- Disk Controller ...

- Is a small processor
- Controls one or more disk drives
- Is capable of:
 - Controlling the actuator that moves the head assembly to a particular cylinder
 - Selecting the block(s) that it wants to read or write.
 - Transfers data between the memory and the selected block.

... -- Disk Controller





-- Disk Storage Characteristics ...

- Some typical measures associated with disk are:
 - Rotation speed: 5400RPM
 - Surfaces = 2 * Platters: 16
 - Tracks per surface $2^{**} 14 = 16384$
 - Bytes per track 1 Mega bytes
 - Sectors per track 256
 - Surface diameter 3.5 inches
 - Sectors per block 4

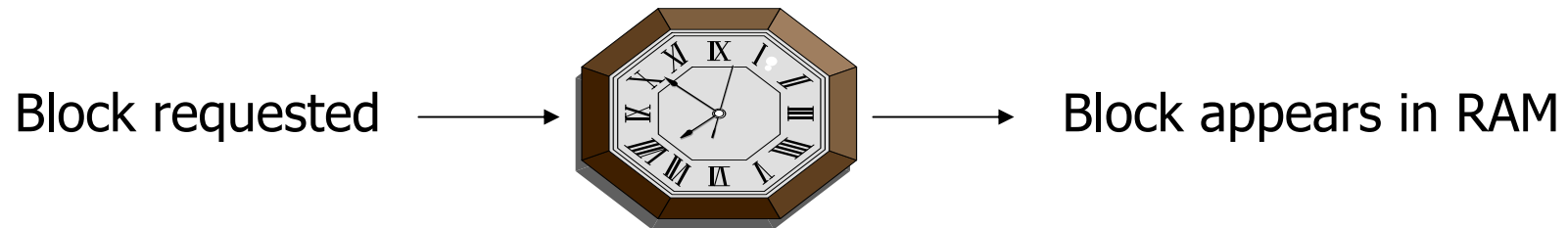


... -- Disk Storage Characteristics

- Capacity = (# bytes/sector) x (avg. # sectors/track) x (# tracks/surface) x (# surfaces/platter) x (# platters/disk)
- Example:
 - 512 bytes/sector
 - 300 sectors/track (on average)
 - 20,000 tracks/surface
 - 2 surfaces/platter
 - 5 platters/disk
- Capacity = $512 \times 300 \times 20000 \times 2 \times 5$
= 30,720,000,000
= 30.72 GB

-- Disk Access Characteristics

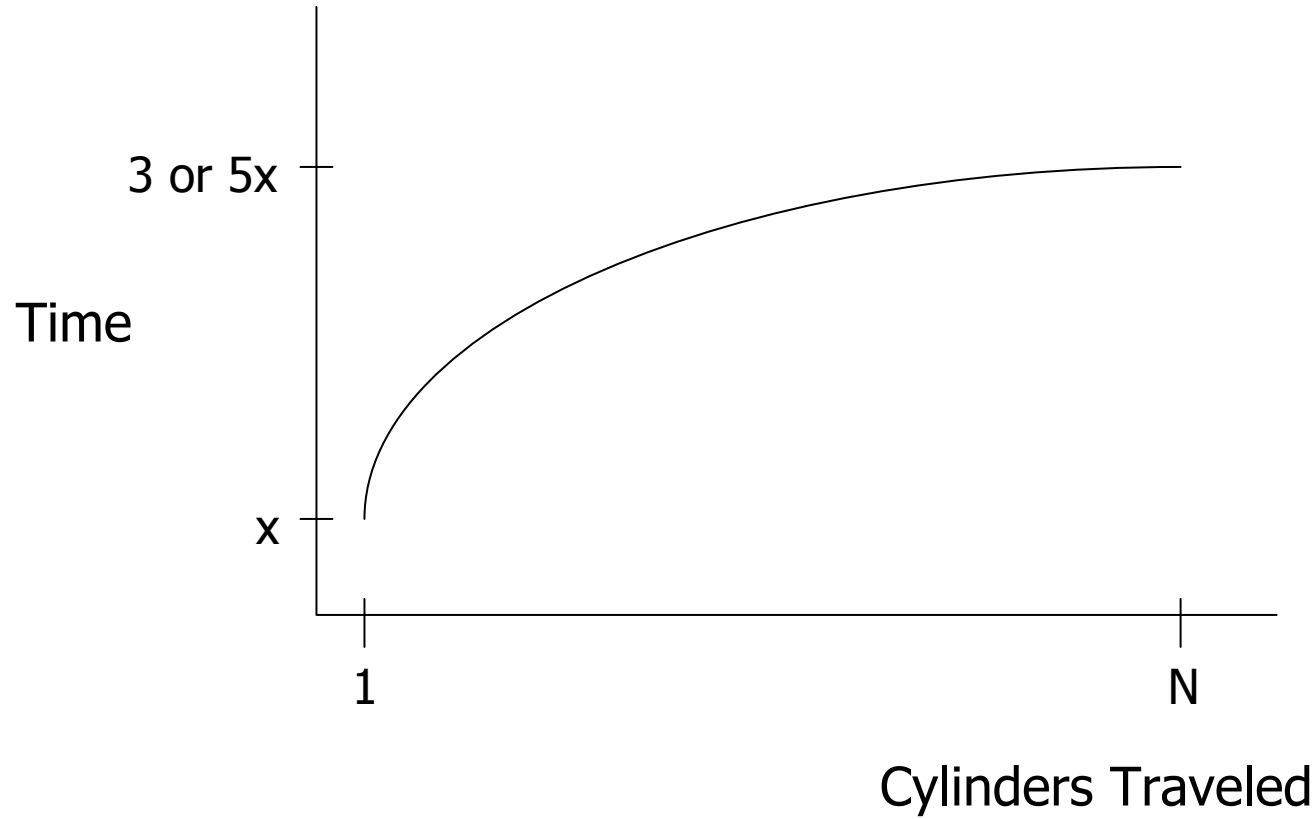
Latency time of disk



Latency of disk = seek time +
rotational latency +
Transfer time



--- Seek Time





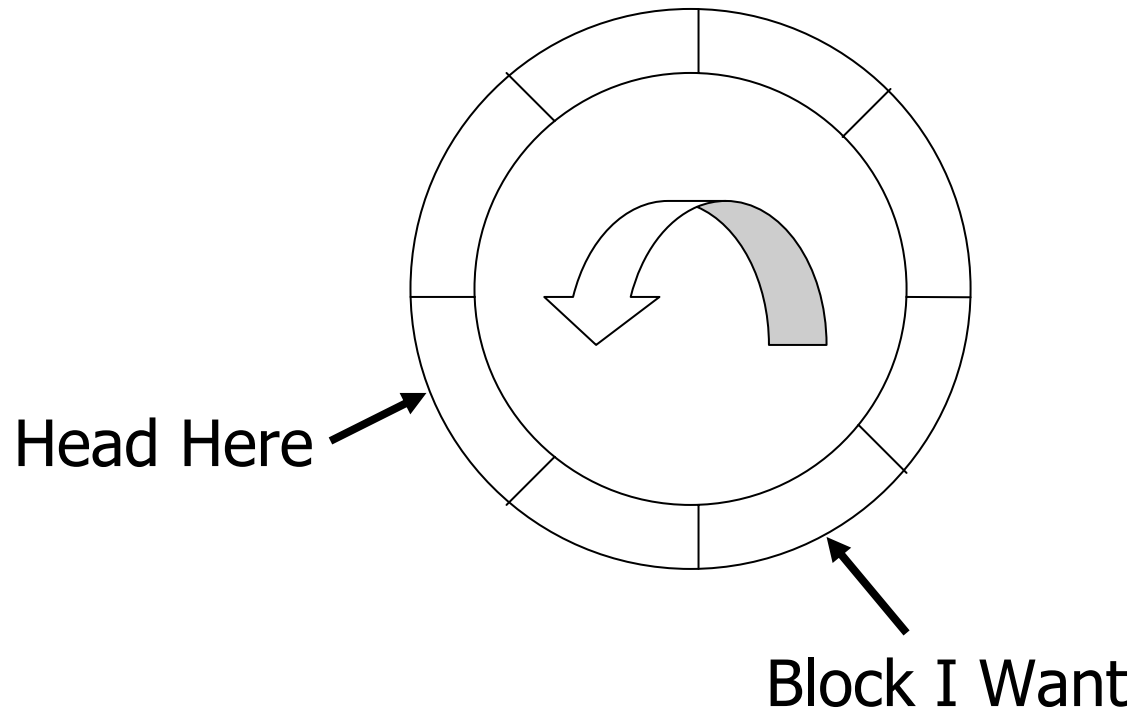
--- Average random seek time

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

"Typical" S : 10 ms \rightarrow 40 ms



--- Rotational Delay





--- Average Rotational Delay

$R = 1/2$ revolution

"typical" $R = 8.33$ ms (3600 RPM)



--- Transfer Rate: t

- “typical” t : 1 \rightarrow 3 MB/second
- transfer time: $\frac{\text{block size}}{t}$



--- Other Delays

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

“Typical” Value: 0



--- Reading next block

- Much cheaper
- Only transfer time: $\frac{\text{block size}}{t}$
- Rule of Thumb
 - Random I/O is expensive than sequential I/O
 - Example:
 - 1 KB block
 - Random I/O $\approx 20\text{ms}$
 - Sequential I/O $\approx 1\text{ms}$



-- Writing a block

- The same cost as reading block
- If you want to verify, you need to add:
 - (full) rotation time + $\frac{\text{Block size}}{t}$



-- Modifying a block

- To modify a block you must do the following:
 1. Read the block in to memory
 2. Make the changes in the memory copy of the block
 3. Write the modified block back to disk
 4. If needed, verify that the write was done correctly.

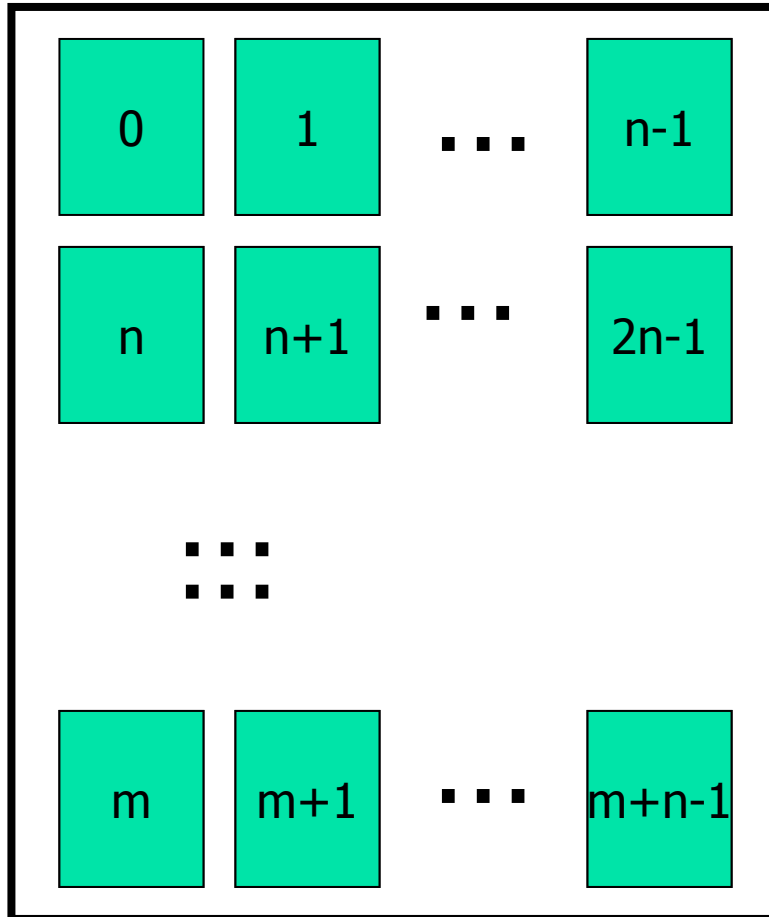


-- Block Address

- Actual block address
 - Physical Device address
 - Cylinder #
 - Surface #
 - Sector #
- Logical block address:
 - mapped by software to 0, 1,2,



--- My Logical Representation of Disks and Blocks





-- Why Read and Write in Blocks ...

- Assume Megatron 747 disk (old)
 - 3.5 in diameter
 - 3600 RPM
 - 1 surface
 - 16 MB usable capacity (16×2^{20})
 - 128 cylinders
 - seek time: average = 25 ms.
adjacent cyl = 5 ms.



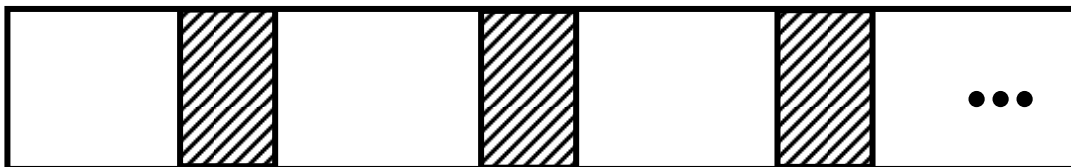
... -- Why Read and Write in Blocks ...

- 1 KB sectors
- 10% overhead between sectors
- capacity = 16 MB = $(2^{20})16 = 2^{24}$
- # cylinders = 128 = 2^7
- bytes/cyl = $2^{24}/2^7 = 2^{17} = 128$ KB
- sectors/cyl = 128 KB / 1 KB = 128

... -- Why Read and Write in Blocks ...

3600 RPM → 60 revolutions / sec
→ 1 rev. = 16.66 msec.

One track:



- Time over useful data: $(16.66)(0.9) = 14.99$ ms.
- Time over gaps: $(16.66)(0.1) = 1.66$ ms.
- Transfer time 1 sector = $14.99/128 = 0.117$ ms.
- Trans. time 1 sector+gap = $16.66/128 = 0.13$ ms.



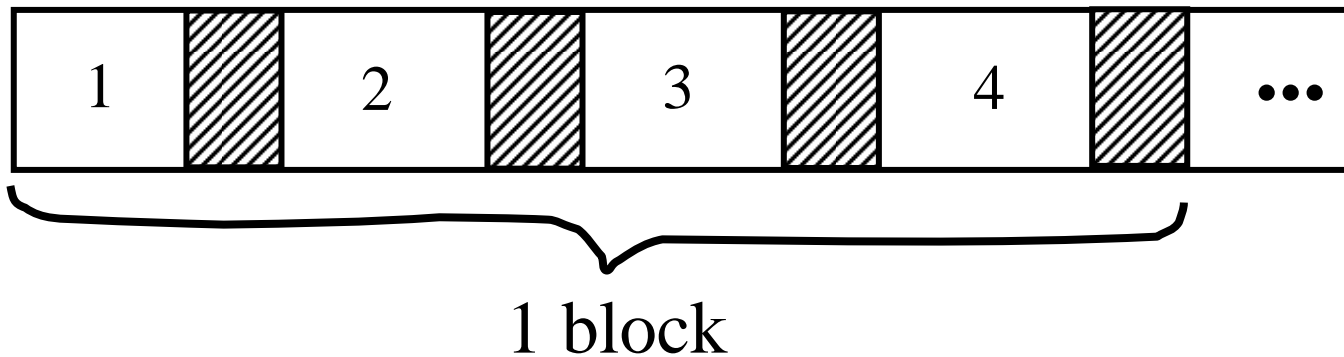
... -- Why Read and Write in Blocks ...

- Time to read one random block

$$\begin{aligned} T_1 &= \text{seek} + \text{rotational delay} + \text{sector transfer time} \\ &= 25 + (16.66/2) + .117 = 33.45 \text{ ms} \end{aligned}$$

... -- Why Read and Write in Blocks ...

- Suppose OS deals with 4 KB blocks



$$T_4 = 25 + (16.66/2) + (.117) \times 1 + (.130) \times 3 = 33.83 \text{ ms}$$

[Compare to $T_1 = 33.45 \text{ ms}$]



- Chapter outline

- The Megatron Database System
- The Memory Hierarchy
- Disks
- **Using Secondary Storage Effectively** ←
- Accelerating Access to Secondary Storage
- Disk Failures
- Recovery from Disk Crashes
- Summary
- References
- Reading List



- Using Secondary Storage Effectively

- RAM Model Algorithms
- The I/O model of Computation
- Sorting Data is Secondary Storage
- Two-Phase, Multiway Merge Sort



-- RAM Model Algorithms

- Algorithms that assume that data is in memory are called RAM model algorithms.
- When implementing DBMS, one should assume that data doesn't fit in to the memory. As a result:
 - RAM model algorithms might not be the best choice.
 - There is a great advantage in using algorithms which use few disk accesses.
- As we will see later, the time of executing a query is approximated by the number of disk I/O done to answer the query.



-- The I/O model of Computation

- For the rest of this lecture let us assume the following model:
 - A computer with one processor, one disk controller and one disk.
 - A database too large to fit in to the main memory.
 - Many concurrent users
 - Disk I/O requests are frequent, the disk controller serves this request in FCFS bases.



-- Sorting Data is Secondary Storage

- Assume
 - A relation **R** of 10,000,000 records is to be sorted
 - Each record of R is **160** bytes.
 - R is stored in a Megatron 747 disk with block size of 16KB.
 - Hence the number records per block:
 - $\text{Floor}(16 * 1024)/160 \approx \mathbf{100}$
 - The size of R in blocks is:
 - $\text{Roof}(10,000,000)/100 = \mathbf{100,000}$
 - If the size of the memory available for R is 100MB, then the number of blocks that can fit into the memory is:
 - $100\text{MB}/16\text{KB} = \mathbf{6400 \text{ blocks}}$
- Is it a good idea to sort R using **quicksort**?



-- Two-Phase, Multiway Merge Sort ...

- The two phase, Multiway Merge sort consists of the following two phase:
 - Phase 1: Creation of sorted runs
 - Phase 2: Merge all the runs (assume N runs)
- Let M denote memory size (in pages).
- Assume
 - 1 memory page = 1 disk block



--- Phase 1

- Create sorted runs.

Let i be 0 initially.

Repeatedly do the following till the end of the relation:

1. Read M blocks of relation into memory
2. Sort the in-memory blocks
3. Write sorted data to run R_i
4. Increment i .

Let the final value of I be N



--- Phase 2:

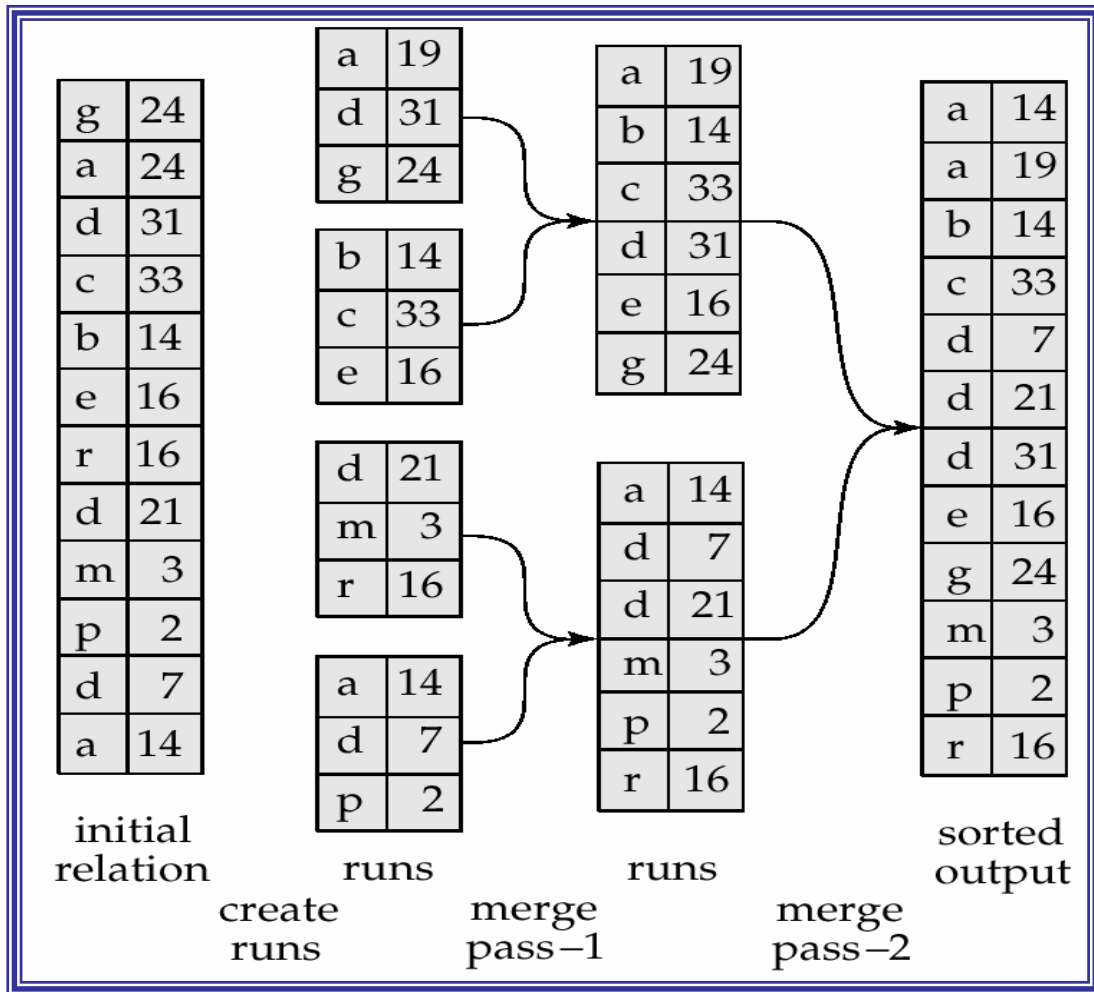
- Merge the runs (N-way merge). We assume (for now) that $N < M$.
 1. Use N blocks of memory to buffer input runs, and 1 block to buffer output. Read the first block of each run into its buffer page
 2. **repeat**
 1. Select the first record (in sort order) among all buffer pages
 2. Write the record to the output buffer. If the output buffer is full write it to disk.
 3. Delete the record from its input buffer page.
If the buffer page becomes empty **then**
 read the next block (if any) of the run into the buffer.
 3. **until** all input buffer pages are empty:



... -- Two-Phase, Multiway Merge Sort ...

- If $N \geq M$, several merge *passes* are required.
 - In each pass, contiguous groups of $M - 1$ runs are merged.
 - A pass reduces the number of runs by a factor of $M - 1$, and creates runs longer by the same factor.
 - E.g. If $M = 11$, and there are 90 runs, one pass reduces the number of runs to 9, each 10 times the size of the initial runs
 - Repeated passes are performed till all runs have been merged into one.

... -- Two-Phase, Multiway Merge Sort ...





... -- Two-Phase, Multiway Merge Sort

- Cost analysis:

- Assume the size of the relation to be sorted is b_r
- Total number of merge passes required: $\lceil \log_{M-1}(b_r/M) \rceil$.
- Disk accesses for initial run creation as well as in each pass is $2b_r$
 - for final pass, we don't count write cost
 - we ignore final write cost for all operations since the output of an operation may be sent to the parent operation without being written to disk

Thus total number of disk accesses for external sorting:

$$b_r (2 \lceil \log_{M-1}(b_r / M) \rceil + 1)$$



- Chapter outline

- The Megatron Database System
- The Memory Hierarchy
- Disks
- Using Secondary Storage Effectively
- **Accelerating Access to Secondary Storage** ←
- Disk Failures
- Recovery from Disk Crashes
- Summary
- References
- Reading List



- Accelerating access to Secondary Storage ...

- The way records are placed in a disk or disks can significantly affect the cost of a query.
- Among some strategies used to decrease query cost by accelerating access to secondary storage are:
 - Organize data by cylinders.
 - Advantage: excellent where access can be predicted in advance, and only one process is using the disk.
 - Disadvantage: No help where access are unpredictable.
 - Using several small disks in place of one large disk.
 - Advantage: Increases the rate at which read/write requests are satisfied.
 - Disadvantage: More \$\$\$\$



... - Accelerating access to Secondary Storage

- Mirror a disks:
 - Advantage:
 - Reliability
 - Parallel access
 - Disadvantage: More \$\$\$\$
- Schedule disk block requests by elevator algorithms
 - Advantage: Minimizes seek time (about $\frac{1}{2}$ block access cost)
 - Disadvantage: Not effective when the number of requests is small.
- Prefetch data in track- or cylinder-sized chunks
 - Advantage: Speed up access when the needed blocks are known
 - Disadvantage:
 - Extra memory
 - No help when access is random.



- Chapter outline

- The Megatron 2002 Database System
- The Memory Hierarchy
- Disks
- Using Secondary Storage Effectively
- Accelerating Access to Secondary Storage
- **Disk Failures** ←
- Recovery from Disk Crashes
- Summary
- References
- Reading List



- Disk Failure

- Types of disk failures:
 - **Intermittent:**
 - Temporary failure of a sector.
 - Solution: repeated reads or writes.
 - **Media decay:**
 - Permanent failure of a sector
 - No solution
 - **Write failure:**
 - Unsuccessful writing to a sector. Possible power outage. Both old and new data is lost.
 - **Disk crash**
- How do you know a sector data is incorrect
 - Checksums. (Ex. parity bits)
- A Solution:
 - Stable storage: Duplicate sectors + checksum



- Chapter outline

- The Megatron 2002 Database System
- The Memory Hierarchy
- Disks
- Using Secondary Storage Effectively
- Accelerating Access to Secondary Storage
- Disk Failures
- **Recovery from Disk Crashes** ←
- Summary
- References
- Reading List



- Recovery From Disk Crash

- Data on the crashed disk can not be accessed (Lost).
- If no backup or no RAID was used, then data can not be recovered.
- The mean time of failure (Time 50% of disks to catastrophically crash) is 10 years for modern disks.
- Solution from disk crash:
 - Use RAID
 - For example Oracle recommends
 - RAID 0 + 1 or
 - RAID 5



- Chapter outline

- The Megatron 2002 Database System
- The Memory Hierarchy
- Disks
- Using Secondary Storage Effectively
- Accelerating Access to Secondary Storage
- Disk Failures
- Recovery from Disk Crashes
- **Summary** ←
- References
- Reading List



- Summary

- The Megatron Database System
- The Memory Hierarchy
- Disks
- Using Secondary Storage Effectively
- Accelerating Access to Secondary Storage
- Disk Failures
- Recovery from Disk Crashes

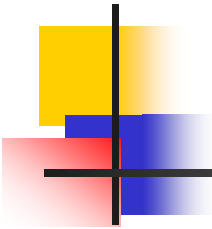


Reading list

- Chapter 11 of your text book



Time for Presentation



Next class is on
Chapter 12,
Representing Data Elements