CS525: Advanced Database Organization

Notes 2: Storage Hardware

Yousef M. Elmehdwi

Department of Computer Science

Illinois Institute of Technology

yelmehdwi@iit.edu

January 10, 2018

Slides: adapted from a courses taught by Hector Garcia-Molina, Stanford, Paris Koutris, & Leonard McMillan

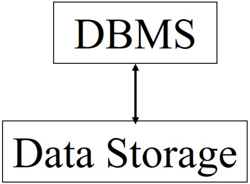
Outline

- Study of data storage in a database management systems
- There are two issues we must address which are related to how a DBMS deals with very large amounts of data efficiently:
  - How does a computer system store and manage very large volumes of data?
  - What representations and data structures best support efficient manipulations of this data?

Today

- Hardware: Disks
- Access Times
- Optimizations
- Other Topics:
    - Storage costs
    - Using secondary storage
    - Disk failures

Hardware



Data Storage

- How does a DBMS store and access data?
  - main memory (fast, temporary)
  - disk (slow, permanent)
- How do we move data from disk to main memory?
  - buffer manager
- How do we organize relational data into files?

Disks and Files

- DBMS stores information on ("hard") disks.
- This has major implications for DBMS design!
  - READ: transfer data from disk to main memory (RAM).
  - WRITE: transfer data from RAM to disk.
  - Both are high-cost operations, relative to in-memory operations, so must be planned carefully!

Why Not Store Everything in Memory?

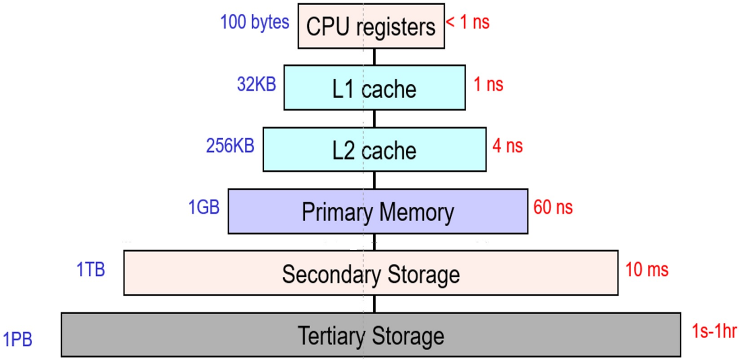
- Relatively high cost
- Main memory is not persistent (volatile)
  - We want data to be saved between runs. (Obviously!)
- Data Size > Memory Size > Address Space

Typical Storage Hierarchy

- CPU Registers temporary variables
- Cash - Fast copies of frequently accessed memory locations
- Main memory (RAM) for currently used "addressable" data.
- Disk for main database (secondary storage)
- Tapes for archiving older versions of the data (tertiary storage)

Memory hierarchy

[1]



Disks

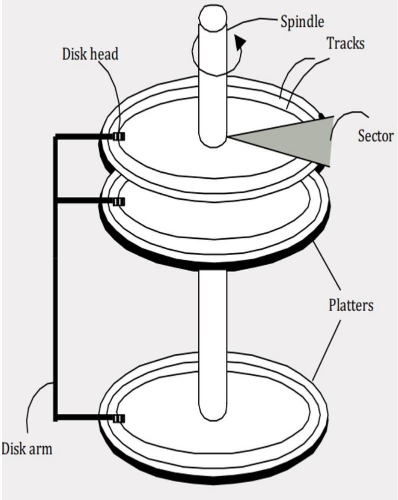
- The use of secondary storage is one of the important characteristics of a DBMS.
- To motivate many of the ideas used in DBMS implementation, we must examine the operation of disks in detail

Disks

- Secondary storage device of choice
- Main advantage over tapes: random access vs. sequential
  - Sequential: read the data contiguously
  - Random: read the data from anywhere at any time
- Data is stored and retrieved in units called disk blocks or pages
  - Typical numbers these days are 64 KB per block
- Retrieval time depends upon the location of the disk
  - Therefore, relative placement of pages on disk has major impact on DBMS performance!

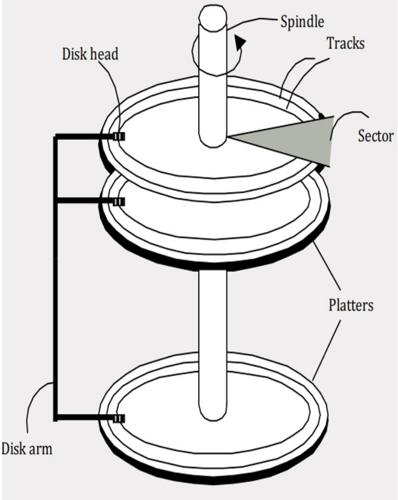
Components of a Disk

- platter: circular hard surface on which data is stored by inducing magnetic changes
- spindle: responsible for rotating the platters
- RPM (Rotations Per Minute): 7200 RPM 15000 RPM

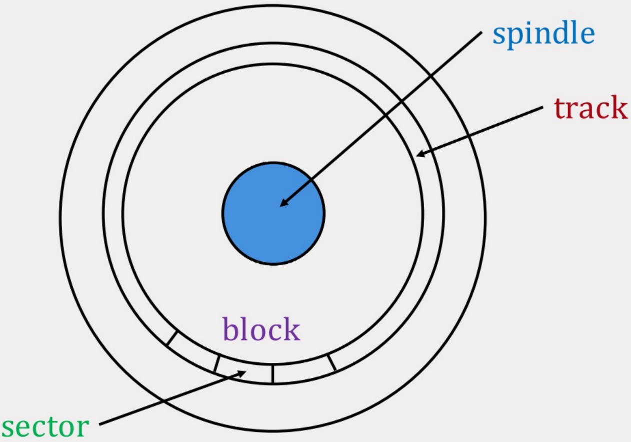


Components of a Disk

- data is encoded in concentric circles of sectors called tracks
- The arm assembly is moved in or out to position a head on a desired track.
- Tracks under heads make a cylinder (imaginary!).
- Only one head reads/writes at any one time.
- Block size is a multiple of sector size (which is fixed).



Top View



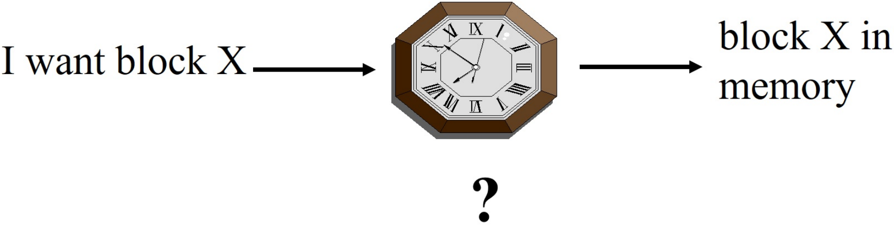
Disk Storage Characteristics

- # Cylinders= # tracks per surface (platter)
- e.g., 10 sectors $\Rightarrow$ 10 cylinders and we can refer to them cylinder zero to cylinder nine
- # tracks per cylinder= # of heads or $2\times$ # platter
- # sector per track
- bytes per sector
- $\Rightarrow$disk capacity/size

Today

- Hardware: Disks
- Access Times
- Optimizations
- Other Topics:
    - Storage costs
    - Using secondary storage
    - Disk failures

Accessing the Disk



- The time taken between the moment at which the command to read a block is issued and the time that the contents of the block appear in main memory is called the latency of the disk.
- The access time is also called the latency of the disk.

Accessing the Disk

Note that blocks can be read or written only when:

- The heads are positioned at the cylinder containing the track on which the block is located, and
- The sectors contained in the block move under the disk head as the entire disk assembly rotates.

Accessing the Disk

access time = seek time + rotational delay + transfer time +other delay

- Other Delays:
  - CPU time to issue I/O
  - Contention for controller
  - Contention for bus, memory
  - “Typical” Value: 0

Accessing the Disk

access time = seek time + rotational delay + transfer time

- Seek time: time to move the arm to position disk head on the right track
- Seek time can be 0 if the heads happen already to be at the proper cylinder.
- If not, the heads require some minimum time to start moving and to stop again, plus additional time that is roughly proportional to the distance traveled.
- The average seek time is often used as a way to characterize the speed of the disk.

Average Random Seek Time

Given N as the total number of tracks

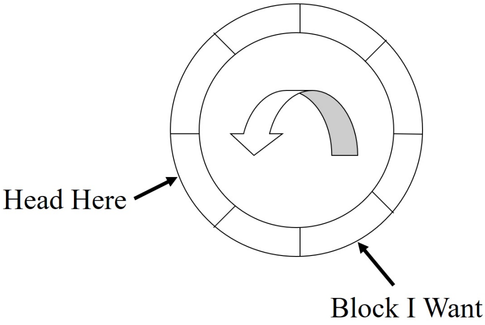
$$S = \frac{\sum_{i=1}^{N} \sum_{j=1, j \neq i}^{N} SEEKTIME(i \rightarrow j)}{N(N-1)}$$

- It sums up all the time traveled between each pair of tracks and get the average of it as the average seek time.

Accessing the Disk

access time = seek time + rotational delay + transfer time

- rotational delay: time to wait for sector to rotate under the disk head



Average Rotational Delay

- On the average, the desired sector will be about half way around the circle when the heads arrive at its cylinder.
- Average rotational delay is time for $\frac{1}{2}$ revolution
- Example: Given a total revolution of 7200 RPM
  - One rotation $= \frac{60s}{7200} = 8.33$ ms
  - Average rotational latency $= 4.16$ ms

Accessing the Disk

access time = seek time + rotational delay + transfer time

- data transfer time: time to move the data to/from the disk surface
- Transfer time is the time it takes the sectors of the block and any gaps between them to rotate past the head.
- Given a transfer rate, the transfer time= $\frac{block\ \ size}{transfer\ \ rate}$

Accessing the Disk

- Seek time and rotational delay dominate.
- Key to lower I/O cost: reduce seek/rotation delays!

Arranging Blocks on Disk

- So far: Random Block Access.
- Blocks in a file should be arranged sequentially on disk (by "next") to minimize seek and rotational delay.
- Next block concept:
  - blocks on same track, followed by
  - blocks on same cylinder, followed by
  - blocks on adjacent cylinder
- For a sequential scan, pre-fetching several blocks at a time is a big win.

If we do things right

- (e.g., Double Buffer, Stagger Blocks)
- Time to get blocks should be proportional to the size of blocks, and the seek time and rotational latency thus become trivial
- time to get block $= \frac{Block\ \ size}{transfer\ \ rate} + Negligible$
- Negligible:
  - skip gap
  - switch track
  - once in a while, next cylinder

Rule of Thumb

- Random I/O: Expensive
- Sequential I/O: Much less

Cost for Writing similar to Reading

- The process of writing a block is, in its simplest form, quite similar to reading a block
- ... unless we want to verify!
- need to add (full) rotation $+\frac{Block\ \ size}{transfer\ \ rate}$

To Modify a Block?

It is not possible to modify a block on disk directly. Rather, even if we wish to modify only a few bytes, we must do the following:

1. Read Block
2. Modify in Memory
3. Write Block
4. [Verify?]

SSD (SOLID STATE DRIVE)

- SSDs use flash memory
- No moving parts (no rotate/seek motors)
  - eliminates seek time and rotational delay
  - very low power and lightweight
- Data transfer rates: 300-600 MB/s
- SSDs can read data (sequential **or** random) very fast!
- Small storage ($0.1 - 0.5\times$ of HDD)
- expensive ($20\times$ of HDD)
- Writes are much more expensive than reads ($10\times$)
- Limited lifetime
  - 1-10K writes per page
  - the average failure rate is 6 years

Today

- Hardware: Disks
- Access Times
- Optimizations
- Other Topics:
  - Storage costs
  - Using secondary storage
  - Disk failures

Optimizations (in controller or O.S.)

Effective ways to speed up disk accesses:

- Disk Scheduling Algorithms
- Track (or larger) Buffer
- Pre-fetch
- Arrays
- Mirrored Disks
- On Disk Cache

Double Buffering

- Another suggestion for speeding up some secondary-memory algorithms is called double buffering.
- In some scenarios, we can predict the order in which blocks will be requested from disk by some process.
- Prefetching (double buffering) is the method of fetching the necessary blocks into the buffer in advance
- Requires enough buffer space
- Speedup factor up to n, where n is the number of blocks requested by a process

Double Buffering

Problem

- Have a File
  - Sequence of Blocks B1, B2
- Have a Program
  - Process B1
  - Process B2
  - Process B3
  - ⋮

Single Buffer Solution

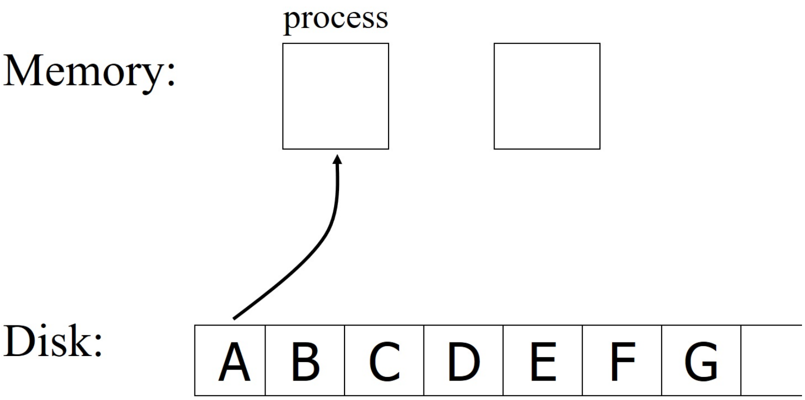
1. Read B1 $\rightarrow$ Buffer
2. Process Data in Buffer
3. Read B2 $\rightarrow$ Buffer
4. Process Data in Buffer
5. $\vdots$

Single Buffer Solution

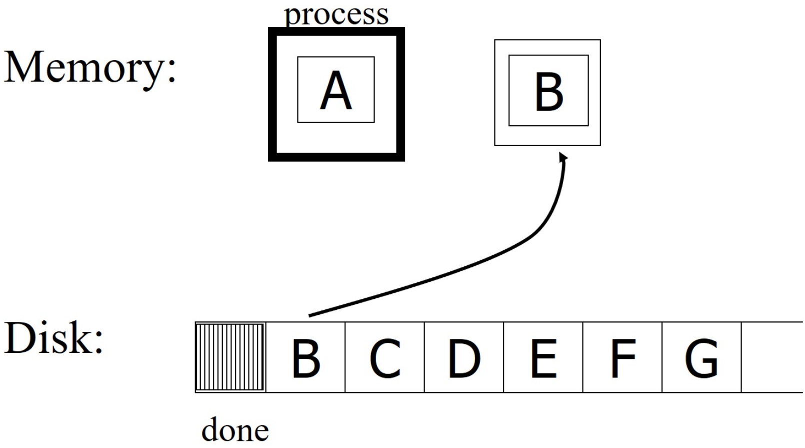
Let:

- P = time to process/block
- R = time to read in 1 block
- n = # blocks
- Single buffer time $= n(P + R)$

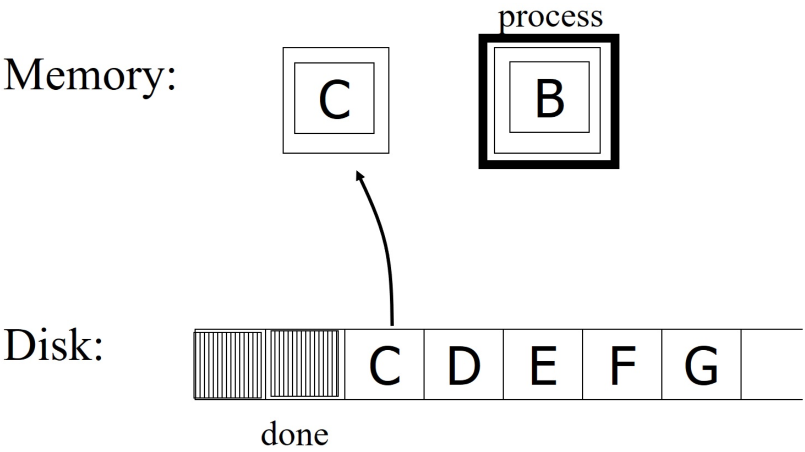
Double Buffering Solution



Double Buffering Solution



Double Buffering Solution



Double Buffer Solution

Let:

- $P =$ time to process/block
- $R =$ time to read in 1 block
- $n = \#$ blocks
- $P \geq R$

What is processing time?

- Double buffering time $= R + nP$
- Single buffer time $= n(R + P)$

Block Size Selection?

- Big Block $\rightarrow$ Less Management Overhead
- Unfortunately
- Big Block $\rightarrow$ Read in more useless stuff and takes longer to read
- Trend: As memory prices drop, blocks get bigger

Disk Failures

We Consider ways in which disks can fail and what can be done to mitigate these failures:

- Partial $\rightarrow$ Total
- Intermittent $\rightarrow$ Permanent

Coping with Disk Failures

- Detection
  - e.g. Checksum
- Correction
  - Redundancy

Megatron 747 Disk (old)

Example

- Rotate at 3600 RPM
- Only 1 surface
- 16 MB usable capacity (usable capacity excludes the gaps)
- 128 cylinders
- seek time:
  - average = 25 ms.
  - adjacent cylinders = 5 ms.
- 1 KB block = 1 sector
- 10% overhead between blocks
  - gaps represent 10% of the circle and
  - sectors represent the remaining 90%

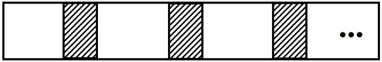
Megatron 747 Disk (old)

- `bytes/cyl` $= \frac{total\ \ capacity}{total\ \#\ cylinders} = \frac{2^{20} \times 16}{128} = \frac{2^{24}}{2^7} = 2^{17} = 128\text{KB}$
- `#blocks/cyl` $= \frac{capacity\ \ of\ \ each\ \ cylinder}{size\ \ of\ \ block} = \frac{128KB}{1KB} = 128$

Megatron 747 Disk (old)

- 3600 RPM $\rightarrow$ 60 revolutions/sec$\rightarrow$1 rev. $=$ 16.66 msec.

One track:



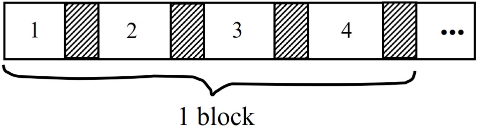
- Time over useful data $= 16.66 \times 0.9 = 14.99$ ms
- Time over gaps$=16.66 \times 0.1 = 1.66$ ms
- Transfer time 1 block $= \frac{14.99}{128} = 0.117$ms
- Q) If there are 128 sectors in each cylinder, then how many gaps are there?
- Transfer time 1 block+gap$= \frac{16.66}{128} = 0.13$ms

Megatron 747 Disk (old)

- T_1 = Time to read one random block
- T_1 = seek + rotational delay + Transfer time 1 block
- $T_1 = 25 + \frac{16.66}{2} + 0.117 = 33.45$ ms.

Megatron 747 Disk (old)

- Suppose OS deals with 4 KB blocks



- $T_4 = 25 + \frac{16.66}{2} + 0.117 \times 1 + 0.13 \times 3 = 33.83$ ms
- Compare to $T_1 = 33.45$ms
- Q) The time to read a full track is ?

Summary

- Secondary storage, mainly disks
- I/O times
- I/Os should be avoided, especially random ones

Reading

- Chapter 2: data storage in `week1/reading` folder, except Sections: 2.3.3, 2.3.4, 2.3.5, 2.4.4, 2.5.4, 2.6

Next

- File and System Structure