Storage

CMPSCI 445

Fall 2008

Slides Courtesy of R. Ramakrishnan and J. Gehrke

Disks and DBMS Design

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

Why Not Store Everything in Main Memory?

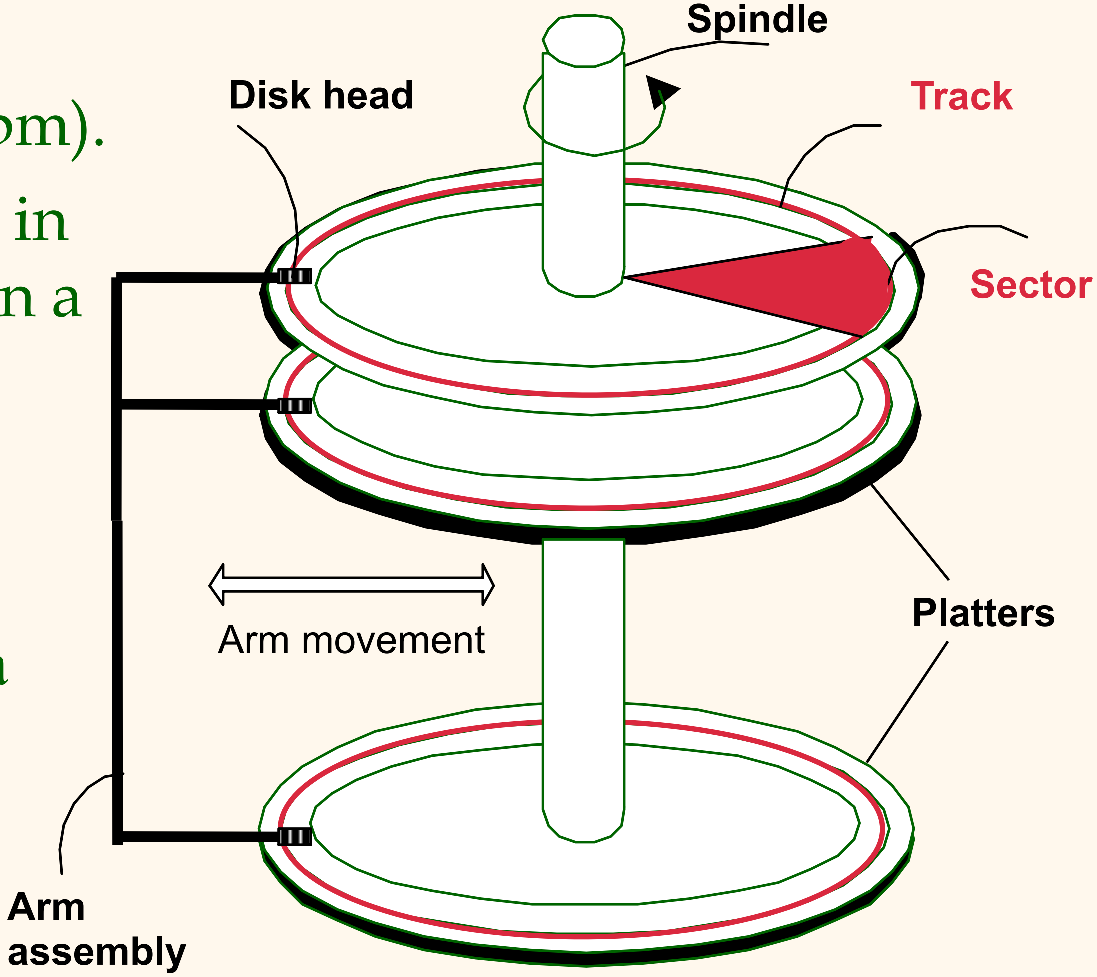
- *Main memory is volatile*. We want data to be saved between runs. (Obviously!)
- *Costs too much*. $100 will buy you either 1GB of RAM or 160GB of disk today.
- *32-bit addressing limitation*.
  - 2^{32} bytes can be directly addressed in memory.
  - Number of objects cannot exceed this number.

Basics of Disks

- Unit of storage and retrieval: *disk block* or *page.*
  - A disk block/page is a contiguous sequence of bytes.
  - Size of a DBMS parameter, 4KB or 8KB.
- Disks support direct access to a page.
- Unlike RAM, time to retrieve a page varies!
  - It depends upon the location on disk.
  - Therefore, relative placement of pages on disk has major impact on DBMS performance!

Components of a Disk

- ❖ Platters spin (say, 7200rpm).
- ❖ Arm assembly is moved in or out to position a head on a desired *track*.
- ❖ Only one head reads/ writes at any one time.



- ❖ Tracks under heads make a *cylinder* (imaginary!).
- ❖ *Each track is divided into sectors (whose size is fixed).*
- ❖ *Block size* is a multiple of *sector size.*

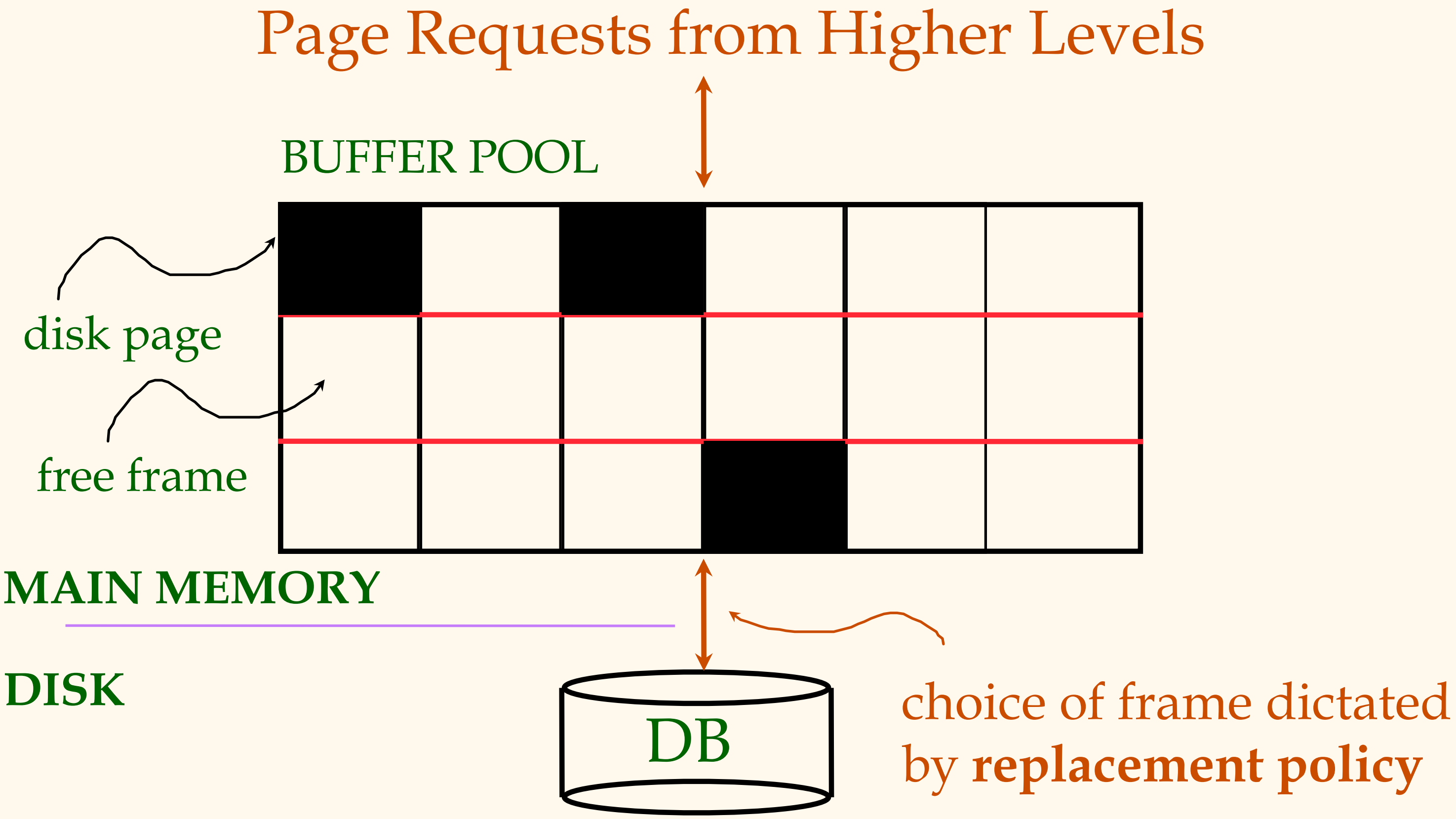
Accessing a Disk Page

- Time to access (read/write) a disk block:
  - *seek time* (moving arms to position disk head on track)
  - *rotational delay* (waiting for block to rotate under head)
  - *transfer time* (actually moving data to/from disk surface)
- Seek time and rotational delay dominate.
  - Seek time varies from about 1 to 20msec
  - Rotational delay varies from 0 to 10msec
  - Transfer rate is about 1msec per 4KB page
- Key to lower I/O cost: reduce seek/rotation delays!

Arranging Pages on Disk

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

Buffer Management in a DBMS


Page Requests from Higher Levels
BUFFER POOL
disk page
free frame
MAIN MEMORY
DISK
DB
choice of frame dictated by **replacement policy**

- *Data must be in RAM for DBMS to operate on it!*
- *Table of <frame#, pageid> pairs is maintained.*

More on Buffer Management

- Requestor of page must unpin it, and indicate whether page has been modified:
  - *dirty* bit is used for this.
- Page in pool may be requested many times,
  - a *pin count* is used. A page is a candidate for replacement iff *pin count* = 0.
- CC & recovery may entail additional I/O when a frame is chosen for replacement. (*Write-Ahead Log* protocol; more later.)

When a Page is Requested ...

- ❖ If requested page is not in pool:
  - ▪ Choose a frame for *replacement*
  - ▪ If frame is dirty, write it to disk
  - ▪ Read requested page into chosen frame
- ❖ *Pin* the page and return its address.

☛ *If requests can be predicted (e.g., sequential scans) pages can be pre-fetched several pages at a time!*

Buffer Replacement Policy

- Frame is chosen for replacement by a *replacement policy:*
  - Least-recently-used (LRU), Clock, MRU etc.
- Policy can have big impact on # of I/O's; depends on the *access pattern*.
- *Sequential flooding*: Nasty situation caused by LRU + repeated sequential scans.
  - # buffer frames < # pages in file means each page request causes an I/O. MRU much better in this situation (but not in all situations, of course).

DBMS vs. OS File System

OS does disk space & buffer mgmt: why not let OS manage these tasks?

- Differences in OS support: portability issues
- Some limitations, e.g., files can't span disks.
- Buffer management in DBMS requires ability to:
  - pin a page in buffer pool, force a page to disk (important for implementing CC & recovery),
  - adjust *replacement policy,* and pre-fetch pages based on access patterns in typical DB operations.