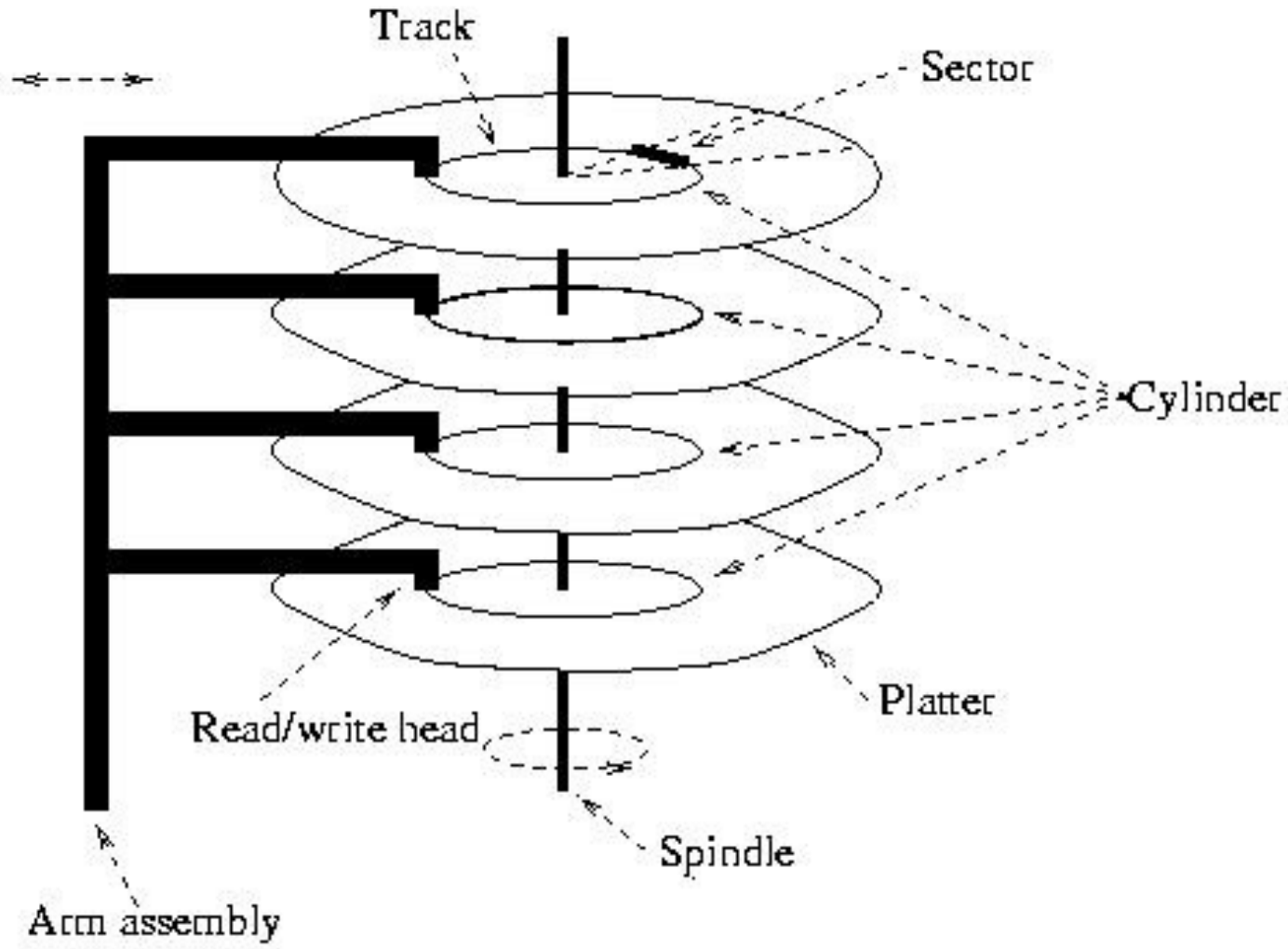
Data Organization on Disks

Davood Rafiei

Physical Disk Structure




DEVON
Login: root Password: abc123
Login: david Password: david123
www.asacomputers.com

Reducing I/O costs

- How crucial is the problem?
  - Disk capacity has improved 1,000X in the last 15 year.
  - The size of data also has increased in the same rate.
  - But,
    - ✓ platters only spin 4X faster.
    - ✓ the transfer rate has improved only 40X in the same period.
- Thus:
  - disk accesses are more precious.
  - they are expected to be more precious in future.

Reducing I/O costs (Cont.)

- Key to lower I/O cost: reduce seek/rotation delays
- Software solutions
  - arrange blocks of a file sequentially on disk
  - read/write in bigger chunks
  - buffering
- Hardware solutions
  - Will not be discussed.

Sequential Access

- Store pages containing related information close together on disk
  - Justification: If application accesses *x,* it will next access data related to *x* with high probability
- Page size tradeoff:
  - Large page size - data related to *x* stored in same page; hence additional page transfer can be avoided
  - Small page size - reduce transfer time, reduce buffer size in main memory
  - Typical page size - 4096 bytes

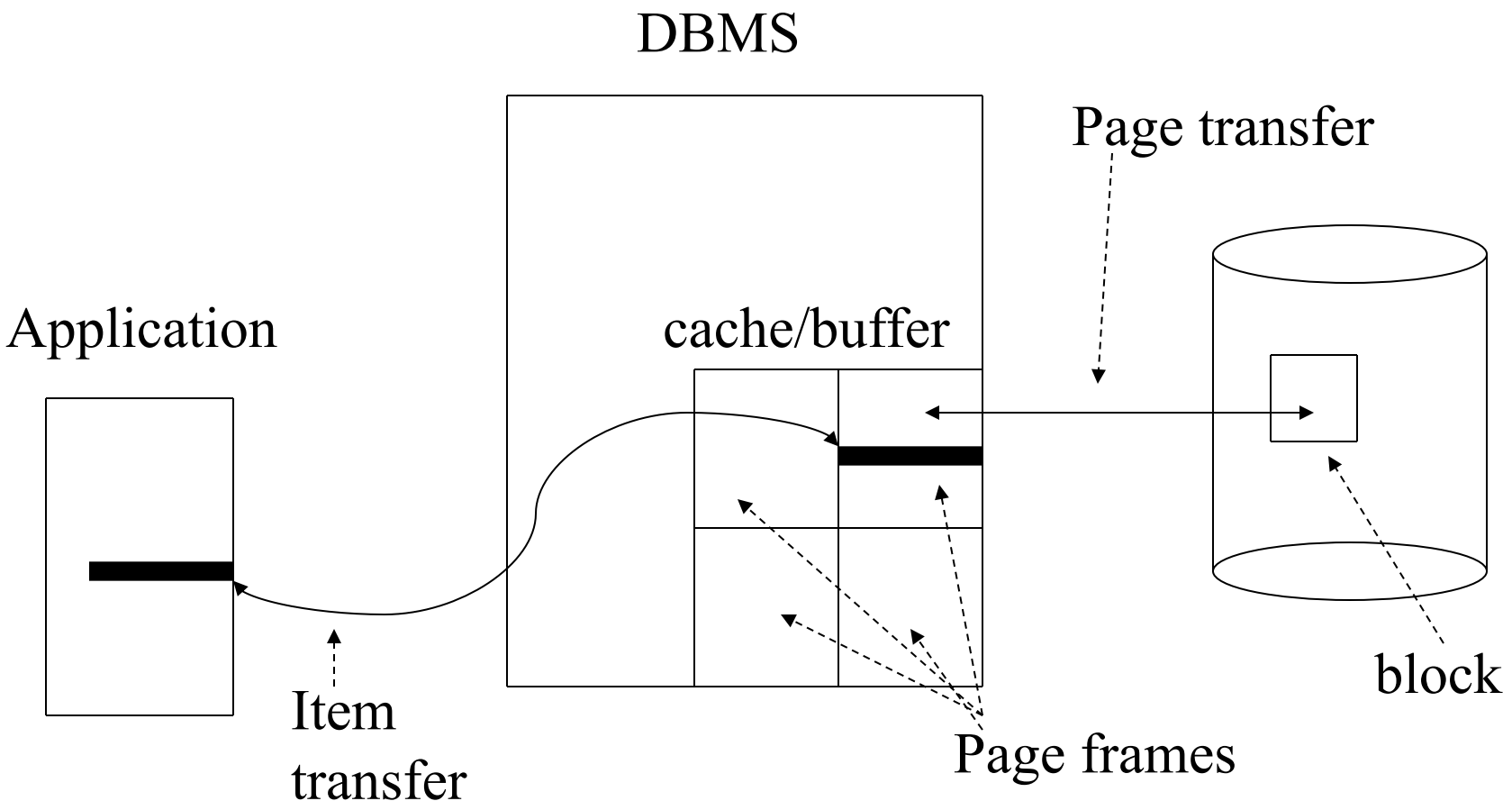
Bigger Chunks

- Consider:
  - An IBM Deskstar disk with 40 sectors/track, 512 bytes/sector and average seek time of 9.1 msec. Disk platters spin at 7,200 rpm.
    - ✓ average rotational delay=(1/7200)/2 minutes = 4.17 msec.
    - ✓ transfer time for a sector= (1/7200)/40 minutes = 0.21 msec
  - A file of 6400 256-byte records = 1638 KB which occupies 3200 sectors of the disk.
- Case 1:
  - The file is stored in 100 extents each of size 4 pages where each page is 8 sectors.
  - Time to read the file= 100 x (9.1 + 4.17 + 32 x 0.21)=2 seconds
- Case 2:
  - The file is stored in 3200 pages each of size one sector.
  - Time to read the file = 3200 ∗ (9.1 + 4.17 + 0.21) = 43 seconds

Buffering

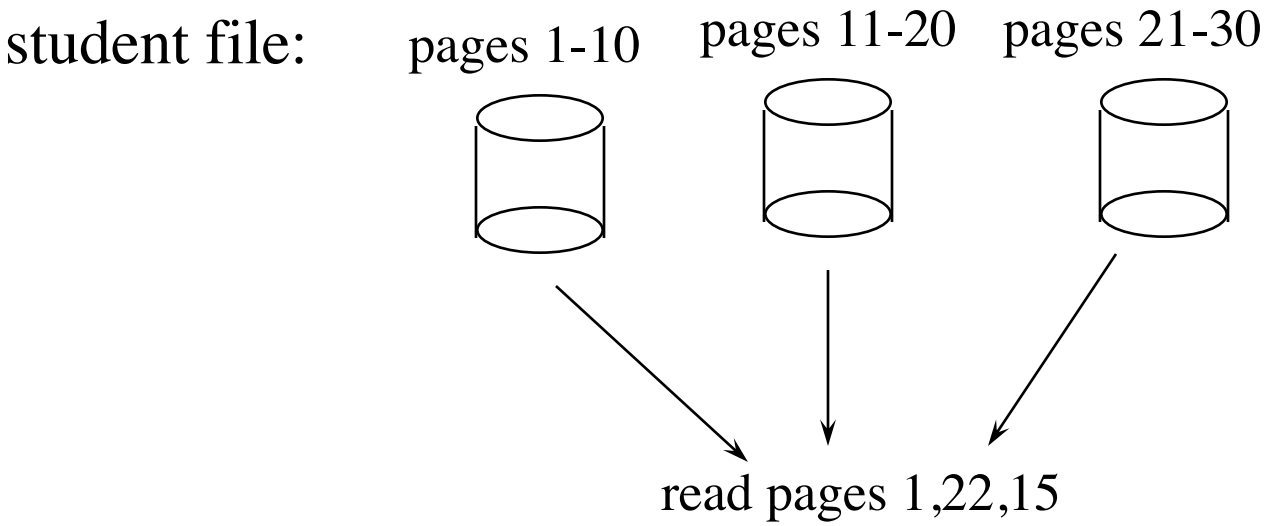
- Keep cache of recently accessed pages in main memory
  - Goal: request for page can be satisfied from cache instead of disk
  - Purge pages when cache is full
    - ✓ For example, use LRU algorithm
    - ✓ Record clean/dirty state of page (clean pages don't have to be written)

Accessing Data Through Cache



Hardware Solutions

- Arrange disks arrays: several disks that give abstraction of a single, large disk.
- Partition data into striping units and distribute them over several disks.



☛ *more disks --> more failures!*

Summary

- Disks: cheap, non-volatile storage.
  - provides both sequential and random access.
  - The cost for a random access depends on the location of page on disk; important to arrange data sequentially to minimize *seek* and *rotation* delays.
- Lowering I/O costs
  - software vs. hardware solutions.