4411

Database Management Systems

Acknowledgements and copyrights: these slides are a result of combination of notes and slides with contributions from: Michael Kiffer, Arthur Bernstein, Philip Lewis, Anestis Toptsis, Addison Wesley, 4411 textbook.

I/O Time to Access a Page

- *Seek time* – time to position heads over cylinder containing page (avg = ~10 - 20 ms)
- *Rotational delay* – additional time for platters to rotate so that start of block containing page is under head (avg = ~5 - 10 ms)
- *Transfer time* – time for platter to rotate over block containing page (depends on size of block)
- *Latency* = seek time + rotational delay
- Seek time and rotational delay dominate.
  - Seek time varies from about 1 to 20msec
  - Rotational delay varies from 0 to 10msec
  - Transfer time is about 1msec per 4KB page

Goal

- minimize latency
- reduce number of page transfers

Reducing Latency …

- Have pages containing related information close together on disk
  - *Justification*: If application accesses item *x,* it will next access data related to *x* with high probability
  - **`*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!

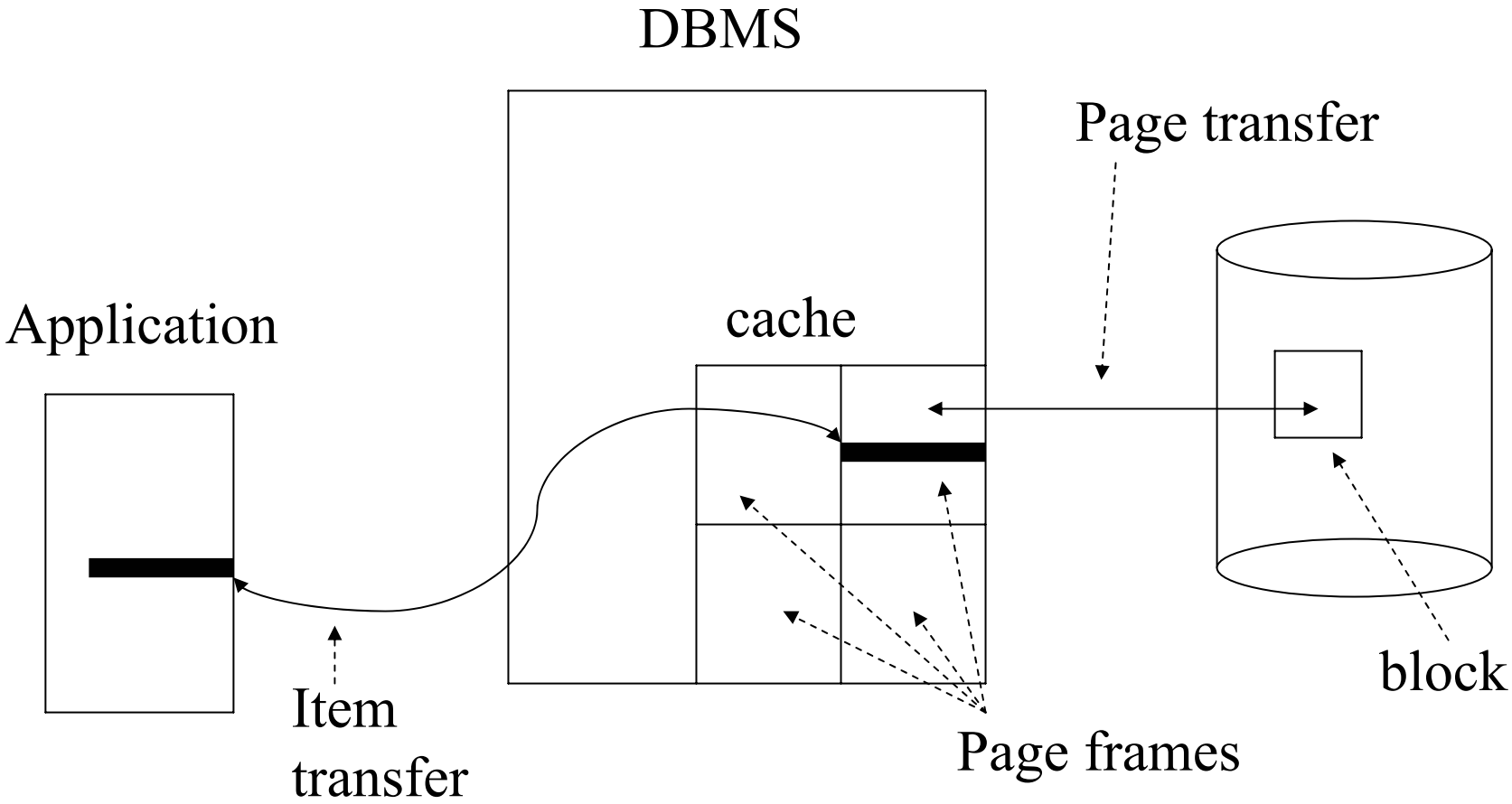
Reducing Latency …/

- 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 – 4 KB or 8 KB

Reducing Number of Page Transfers

- Keep cache of recently accessed pages in main memory
  - *Rationale*: 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



RAID Systems

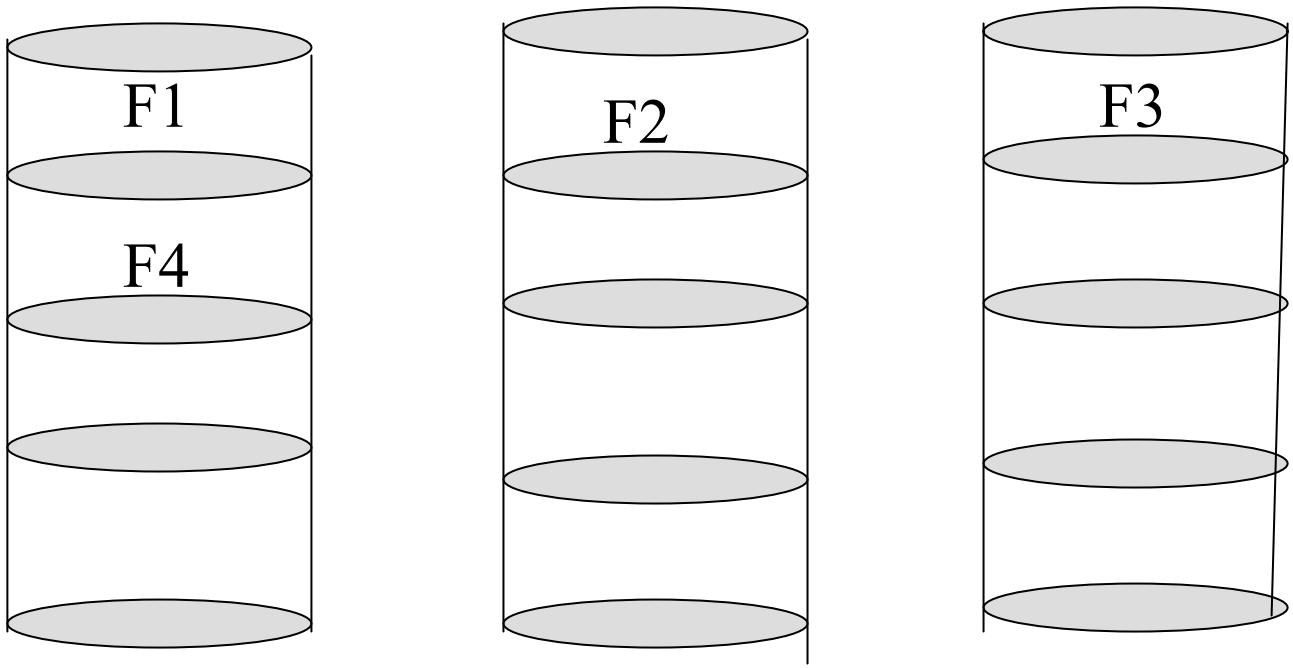
- RAID (Redundant Array of Independent Disks) is an array of disks configured to behave like a single disk with
  - Higher throughput (increased speed)
    - Multiple requests to different disks can be handled independently
    - If a single request accesses data that is stored separately on different disks, that data can be transferred in parallel
  - Increased reliability
    - Data is stored redundantly
    - If one disk should fail, the system can still operate

RAID Systems

- Two main techniques:
  - Data Striping: Data is partitioned; Partitions are distributed over several disks.
  - Data Mirroring (Redundancy): Data is duplicated and each duplicate is stored on a different disk (mirror disk). Redundant information allows reconstruction of data if a disk fails. (however, More disks => more failures.).

Striping

- Data that is to be stored on multiple disks is said to be *striped*
  - Data is divided into *chunks*
    - Chunks might be bytes, disk blocks etc.
  - If a file is to be stored on three disks, in a round-robin fashion
    - First chunk is stored on first disk
    - Second chunk is stored on second disk
    - Third chunk is stored on third disk
    - Fourth chunk is stored on first disk
    - And so on



The striping of a file partitioned into 4 chunks, across 3 disks

Levels of RAID System

- **Level 0**: Striping but no redundancy
  - A striped array of *n* disks
  - The failure of a single disk ruins everything

RAID Levels …

- **Level 1**: Mirrored Disks (no striping)
  - An array of ***2*** mirrored disks
    - All data stored on two disks
    - Each disk holds an identical copy of the other disk.
  - Increases reliability
    - If one disk fails, the system can continue
  - Increases speed of reads
    - Both of the mirrored disks can be read concurrently
  - Decreases speed of writes
    - Each write must be made to two disks
  - Requires twice the number of disks

RAID Levels …

- **Level 10**: A combination of levels 0 and 1 (not an official level)
  - A striped array of n disks (as in level 0)
  - Each of these disks is mirrored (as in level 1)
    - Achieves best performance of all levels
    - Requires twice as many disks

RAID Levels (con't)

- **Level 3**: Data is striped over n disks and an $(n+1)^{th}$ disk is used to store the exclusive or (XOR) of the corresponding bytes on the other n disks
  - The $(n+1)^{th}$ disk is called the parity disk
  - Chunks are bytes

Level 3 (con't)

- Redundancy increases reliability
  - Setting a bit on the parity disk to be the XOR of the bits on the other disks makes the corresponding bit on each disk the XOR of the bits on all the other disks, including the parity disk

    1 0 1 0 1 1 (parity disk)
  - If any disk fails, its information can be reconstructed as the XOR of the information on all the other disks

Level 3 (con't)

- Whenever a write is made to any disk, a write must by made to the parity disk

  New_Parity_Bit = Old_Parity_Bit XOR
  (Old_Data_Bit XOR New_Data_Bit)

- Thus each write requires 4 disk accesses
- The parity disk can be a bottleneck since all writes involve a read and a write to the parity disk