Principles of Data Management

Lecture #2
(Storing Data: Disks and Files)

Instructor: Mike Carey
mjcarey@ics.uci.edu

Today’s Topics

- Today is the real start of this course!
- Remember to read the course wiki page
  - http://www.ics.uci.edu/~cs222
  - http://www.ics.uci.edu/~cs122c
- Also follow the Piazza page
  - https://piazza.com/uci/fall2014/cs122ccs222/home
  - There have already been lots of good questions and answers appearing there...
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 Main Memory?

- **Costs too much**. Dell wants (in early 2014) $65 for 500GB of disk, $600 for 256GB of SSD, and $57 for 4GB of RAM ($0.13, $2.34, $14.25 per GB)
- **Main memory is volatile**. We want data to be saved between runs. (Obviously!)
- Your typical (basic) storage hierarchy:
  - Main memory (RAM) for currently used data
  - Disk for the main database (secondary storage)
  - Tapes for archiving older versions of the data (tertiary storage)
- And we also have L1 & L2 caches, SSD, …
**Storage Hierarchy & Latency (Jim Gray): How Far Away is the Data?**

<table>
<thead>
<tr>
<th>Storage Level</th>
<th>Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tape / Optical Robot</td>
<td>$10^9$ years</td>
</tr>
<tr>
<td>Disk</td>
<td>$10^6$ years</td>
</tr>
<tr>
<td>Memory</td>
<td>1.5 hours</td>
</tr>
<tr>
<td>On Chip Cache</td>
<td>10 minutes</td>
</tr>
<tr>
<td>Registers</td>
<td>1 minute</td>
</tr>
<tr>
<td>This Room</td>
<td></td>
</tr>
<tr>
<td>This Building</td>
<td></td>
</tr>
<tr>
<td>La Jolla (UCSD)</td>
<td></td>
</tr>
<tr>
<td>Andromeda</td>
<td>2,000 years</td>
</tr>
<tr>
<td>Pluto</td>
<td>2 years</td>
</tr>
<tr>
<td>My Head</td>
<td></td>
</tr>
<tr>
<td>Pluto</td>
<td>2,000 years</td>
</tr>
<tr>
<td>Andromeda</td>
<td>2 years</td>
</tr>
</tbody>
</table>

**Disks**

- Secondary storage device of choice.
- Main advantage over tapes: *random access* vs. *sequential*.
- Data is stored and retrieved in units called *disk blocks* or *pages*.
- Unlike RAM, time to retrieve a disk page varies depending upon location on disk.
  - Therefore, relative placement of pages on disk has a *major* impact on DBMS performance!
  - (SSDs simplify things a bit in this respect)
Components of a Disk

- The platters spin (5400 rpm)
- The arm assembly is moved in or out to position a head on a desired track
- Tracks under heads form a cylinder (imaginary!)
- Only one head reads/writes at any one time.
- Block size is a multiple of sector size (which is fixed)

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 1 msec per 4KB page (old)
- Key to lower I/O cost: Reduce seek/rotation delays! Hardware vs. software solutions?
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`) in order to minimize seek and rotational delay
- For a sequential scan, prefetching several pages at a time is a big win!

RAID (Redundant Array of Inexpensive Disks)

- Disk Array: Arrangement of several disks that gives abstraction of a single, large disk.
- Goals: Increase performance and reliability.
- Two main techniques:
  - Data striping: Data is partitioned; size of a partition is called the striping unit. Partitions are distributed over several disks.
  - Redundancy: More disks => more failures. Redundant information allows reconstruction of data if a disk fails.
**RAID Levels**

- **Level 0**: No redundancy (just striping)
- **Level 1**: Mirrored (two identical copies)
  - Each disk has an exact mirror image
  - Parallel reads; writes involve two disks
  - Maximum transfer rate = transfer rate of one disk
- **Level 0+1**: Striping and Mirroring
  - Parallel reads; writes involve two disks
  - Maximum transfer rate = aggregate bandwidth

**RAID Levels (Contd.)**

- **Level 3**: Bit-Interleaved Parity
  - Striping Unit: One bit. (One check disk.)
  - Each read and write request involves all disks; disk array can process one request at a time.
- **Level 4**: Block-Interleaved Parity
  - Striping Unit: One disk block. (One check disk.)
  - Parallel reads possible for small requests, large requests can utilize full bandwidth
  - Writes involve modified block and check disk
- **Level 5**: Block-Interleaved Distributed Parity
  - Similar to RAID Level 4, but parity blocks are distributed over all disks
Disk Space Management

- Lowest layer of DBMS software manages the space on disk.
- Higher levels call upon this layer to:
  - allocate/de-allocate a page
  - read/write a page
- A request for a sequence of pages must be satisfied by allocating the pages sequentially on disk! Higher levels don’t need to know how this is done or how free space is managed.
Buffer Management in a DBMS

Page Requests from Higher Levels

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

When a Page is Requested ...

- If requested page is not in pool:
  - Choose a frame for replacement
  - If that 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 prefetched several pages at a time!
More on Buffer Management

- Requestor of page must unpin it, and indicate whether page has been modified, when done:
  - dirty bit used for the latter purpose
- Page in pool may be requested many times
  - a pin count is used, and 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 in CS 223.)

Buffer Replacement Policy

- Frame is chosen for replacement using 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 management – so why not let the 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), and
  - adjust replacement policy, and prefetch pages based on access patterns in typical DB operations.

**Next Time…**

- We’ll talk about files of records and associated approaches to
  - Space management
  - Record formats
  - File formats
- You should read ahead – after we’ve covered this stuff you will know everything needed for Project 1 (and you already know most of it…) – and you should have gotten started on Project 1 already (and many of you clearly have)