Lecture 9:
Data Storage and IO Models
Announcements

• Submission Project Part 1 tonight
  • Instructions on Piazza!

• PS2 due on Friday at 11:59 pm
  • Questions? Easier than PS1.

• Badgers Rule!
Today’s Lecture

1. Data Storage

2. Disk and Files

3. Buffer Manager - Prelims
1. Data Storage
What you will learn about in this section

1. Life cycle of a query

2. Architecture of a DBMS

3. Memory Hierarchy
Life cycle of a query

Database Server

- Parser
- Optimizer
- Query Scheduler
- Execute Operators

Query

Query Result

Syntax Tree
Query Plan
Segments
Internal Architecture of a DBMS

- Query Execution
  - Data access
    - Storage Manager
      - I/O access
Architecture of a Storage Manager

Access Methods
- Sorted File
- Heap File
- Hash Index
- B+-tree Index

Concurrency Control Manager

Recovery Manager

I/O Manager

Buffer Manager

IO Accesses

In Systems, IO cost matters a ton!
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?
Memory Hierarchy

- CPU
- Cache
- Main Memory
- Flash Storage
- Magnetic Hard Disk Drive (HDD)

Access Speed: $10^{-8} - 10^{-6}$ cycles

Capacity: $10^5 - 10^6$ cycles

Price: $10^7 - 10^8$ cycles
Why not main memory?

- Relatively high cost
- Main memory is not persistent!

Typical storage hierarchy:
- **Primary storage**: main memory (RAM) for currently used data
- **Secondary storage**: disk for the main database
- **Tertiary storage**: tapes for archiving older versions of the data
2. Disk and Files
What you will learn about in this section

1. All about disks
2. Accessing a disk
3. Managing disk space
Disks

• Secondary storage device of choice.

• Data is stored and retrieved in units called *disk blocks*

• Unlike RAM, time to retrieve a disk page varies depending upon location on disk.
  
  • Therefore, relative placement of pages on disk has major impact on DBMS performance!
The Mechanics of a Disk

Mechanical characteristics:
• Rotation speed (7200RPM)
• Number of platters (1-30)
• Number of tracks (<=10000)
• Number of bytes/track($10^5$)
The Mechanics of a Disk

- Platters spin @ ~ 7200rpm
- Arm assembly moves to position a head on a desired track. Tracks under heads make a **cylinder** (imaginary!)
- Only 1 head reads/writes at any time
- **Block size**: multiple of sector size (which is fixed).
The Mechanics of a Disk

Unit of read or write: 
**disk block: k*Sector Size**

Once in memory: **page**
Typically: 4k or 8k or 16k

**Access time** = seek time + rotational delay + transfer time
(1-20 ms)  (0-10ms)  (~1 ms per 8k block)
The Mechanics of a Disk

GOAL: Minimize seek and rotational delay

“Next Block” concept
(1) Blocks on same track
(2) Blocks on same cylinder
(3) Blocks on adjacent cylinder

Disks read/write one block at a time

Access time = seek time + rotational delay + transfer time

For a sequential scan, pre-fetching several pages at a time is a big win!
Accessing the disk (I)

**access time** = \textit{rotational delay} + \textit{seek time} + \textit{transfer time}

\textbf{rotational delay}: time to wait for sector to rotate under the disk head

- typical delay: 0–10 ms
- maximum delay = 1 full rotation
- average delay ~ half rotation

\begin{center}
\begin{tabular}{|c|c|}
\hline
RPM & Average delay \\
\hline
5,400 & 5.56 \\
7,200 & 4.17 \\
10,000 & 3.00 \\
15,000 & 2.00 \\
\hline
\end{tabular}
\end{center}
Accessing the disk (II)

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

**seek time**: time to move the arm to position disk head on the right track
- typical seek time: $\sim 9 \text{ ms}$
- $\sim 4 \text{ ms}$ for high-end disks
Accessing the disk (III)

access time = rotational delay + seek time + transfer time

data transfer time: time to move the data to/from the disk surface

• typical rates: \(~100\ MB/s\)

• the access time is dominated by the seek time and rotational delay!
Example: Specs

<table>
<thead>
<tr>
<th></th>
<th>Seagate HDD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capacity</td>
<td>3 TB</td>
</tr>
<tr>
<td>RPM</td>
<td>7,200</td>
</tr>
<tr>
<td>Average Seek Time</td>
<td>9 ms</td>
</tr>
<tr>
<td>Max Transfer Rate</td>
<td>210 MB/s</td>
</tr>
<tr>
<td># Platters</td>
<td>3</td>
</tr>
</tbody>
</table>

What are the I/O rates for block size 4 KB and:

• random workload (~ 0.3 MB/s)
• sequential workload (~ 210 MB/s)
Managing Disk Space

• The disk space is organized into files
• Files are made up of pages
• Pages contain records

• Data is allocated/deallocated in increments of pages
• Logically close pages should be nearby in the disk
SSDs (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!
SSDs

- Small storage (0.1-0.5x of HDD)
- Expensive (20x of HDD)
- **Writes** are much more expensive than **reads** (10x)
- Limited lifetime
  - 1-10K writes per page
  - The average failure rate is 6 years

Can only read and write in blocks or pages of 2K, 4K, or more bytes. Looks like a disk.
3. Buffer Manager - Prelims
What you will learn about in this section

1. Buffer Manager

2. Replacement Policy
High-level: Disk vs. Main Memory

Disk:

- **Slow**: Sequential block access
  - Read a blocks (not byte) at a time, so sequential access is cheaper than random
  - Disk read / writes are expensive!

- **Durable**: We will assume that once on disk, data is safe!

- **Cheap**

Random Access Memory (RAM) or Main Memory:

- **Fast**: Random access, byte addressable
  - ~10x faster for sequential access
  - ~100,000x faster for random access!

- **Volatile**: Data can be lost if e.g. crash occurs, power goes out, etc!

- **Expensive**: For $100, get 16GB of RAM vs. 2TB of disk!
The Buffer

- A **buffer** is a region of physical memory used to store *temporary data*

  - *In this lecture*: a region in main memory used to store *intermediate data between disk and processes*

- **Key idea**: Reading / writing to disk is slow - need to cache data!
The (Simplified) Buffer

- In this class: We’ll consider a buffer located in **main memory** that operates over **pages** and **files**:

  - **Read**(page): Read page from disk -> buffer *if not already in buffer*
The (Simplified) Buffer

• In this class: We’ll consider a buffer located in **main memory** that operates over **pages** and **files**:

  • **Read(page):** Read page from disk -> buffer *if not already in buffer*

  Processes can then read from / write to the page in the buffer
The (Simplified) Buffer

- In this class: We’ll consider a buffer located in **main memory** that operates over **pages** and **files**:
  - **Read(page):** Read page from disk -> buffer *if not already in buffer*
  - **Flush(page):** Evict page from buffer & write to disk
The (Simplified) Buffer

• In this class: We’ll consider a buffer located in main memory that operates over pages and files:
  
  • **Read(page):** Read page from disk -> buffer *if not already in buffer*
  
  • **Flush(page):** Evict page from buffer & write to disk
  
  • **Release(page):** Evict page from buffer *without* writing to disk
Managing Disk: The DBMS Buffer

- Database maintains its own buffer
  - Why? The OS already does this...
  - DB knows more about access patterns.
    - Watch for how this shows up! (cf. Sequential Flooding)
  - Recovery and logging require ability to flush to disk.
The Buffer Manager

• A buffer manager handles supporting operations for the buffer:
  
  • Primarily, handles & executes the “replacement policy”
    • i.e. finds a page in buffer to flush/release if buffer is full and a new page needs to be read in
  
  • DBMSs typically implement their own buffer management routines
A Simplified Filesystem Model

• For us, a **page** is a *fixed-sized array* of memory
  • Think: One or more disk blocks
  • Interface:
    • write to an entry (called a **slot**) or set to “None”

• DBMS also needs to handle variable length fields
  • Page layout is important for good hardware utilization as well (see 346)

• And a **file** is a *variable-length list* of pages
  • Interface: create / open / close; next_page(); etc.
Buffer Manager

- Data must be in RAM for DBMS to operate on it
- All the pages may not fit into main memory

**Buffer manager**: responsible for bringing pages from disk to main memory as needed. Pages brought into main memory are in the **buffer pool**. The buffer pool is partitioned into **frames**: slots for holding disk pages.
Buffer Manager

buffer pool

page request

page

frame

disk
Remember: Buffer Manager

• **Read**(*page*): Read page from disk -> buffer *if not already in buffer*

• **Flush**(*page*): Evict page from buffer & write to disk

• **Release**(*page*): Evict page from buffer *without* writing to disk
Buffer replacement policy

- How do we choose a frame for replacement?
  - LRU (Least Recently Used)
  - Clock
  - MRU (Most Recently Used)
  - FIFO, random, ...

- The replacement policy has big impact on # of I/O’s (depends on the access pattern)

- To be continued!