Principles of Database Management Systems

Pekka Kilpeläinen
(after Stanford CS245 slide originals by Hector Garcia-Molina, Jeff Ullman and Jennifer Widom)

Outline
- Hardware: Disks
- Access Times
- Example - Megatron 747
- External sorting
- Other Topics:
  - Optimizing disk usage

Memory Hierarchy: Capacities and access times
- Terabytes: 10-100 s
- Gigabytes: 10 ms = 10^{-2} s
- Megabytes: 100 ns = 10^{-7} s

CPU vs. Disk Speed
- CPU: 100 → 500 → 1000 MIPS
  - Main memory access 10^{-6} → 10^{-9} sec.
- DISK read/write approx. 10 ... 30 ms
- CPU executes roughly 10^{6} instructions in the time of a disk access
  - disk access times do not shrink in proportion to CPU speed-up
- → disk I/O major DBMS concern

Typical Disk
1..n rotating platters (of two surfaces); Surfaces accessed by heads (moving together), divided into concentric tracks. Tracks under the heads at the same time form a cylinder. Tracks are divided into sectors separated by gaps.
Top View: tracks on a surface

“Typical” Numbers
Diameter: 1 ... 3.5 ... 15 ”
Cylinders: 100 ... 2000 (floppy: 40)
Surfaces: 1 (CD, old floppy) →
(=tracks/cyl) 2 (floppy, DVD) → 30
Sector Size: 0.5 ... 4KB ... 50KB
Capacity: 1.44 MB (floppy)
... several GBs

Physical vs. Logical I/O
• Sector: indivisible unit of disk I/O
• Blocks
  - logical (DBMS/OS) units of disk storage
  - transferred btw disk and main memory buffers
  - typically 4...56 KB, consisting of one or more sectors

Disk Access Time
I want block X in memory

Time = Seek Time + Rotational Delay + Transfer Time + Other

Seek Time $S$

Real seek time
Simplified approximation

Cylinders Traveled
**Average Random Seek Time** $E(S)$

$$E(S) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{N} \sum_{j=1}^{N} \text{SEEKTIME (i → j)}$$

$= (\text{approx.}) \frac{1}{3} \text{SEEKTIME (1 → N)}$

“Typical” $E(S)$: 3 ... 15 ms

---

**Rotational Delay** $R$

- Head Here
- Block I Want

---

**Average Rotational Delay** $E(R)$

$$E(R) = \frac{1}{2} \times \text{full rotation time}$$

“Typical” rotation speed 3600 RPM

$\rightarrow$ full rotation in $1/60$ s

$\rightarrow E(R) = 1/120$ s = 8.33 ms

---

**Transfer Rate**: $t$

- “typical” $t$: 1 ... 3 MB/second
- transfer time: $(\text{block size})/t$

**Other Delays**:
- CPU time to issue I/O
- Contention for controller
- Contention for bus, memory

“Typical” Value $\approx 0$

---

So far: Random Block Access

- What about: Reading “Next” block?

---

If we do things right (e.g., Arrange by Cylinders, Double Buffer ...)

Time to get block = $\frac{\text{Block Size}}{t} + \text{Negligible}$

- skip gap
- switch track
- once in a while, next cylinder
**Rule of Thumb**

Random I/O: Expensive  
Sequential I/O: Much less

- Ex: 1 KB Block  
  »Random I/O: ~ 20 ms.  
  »Sequential I/O: ~ 1 ms.

Cost for **Writing** similar to **Reading**

... unless we want to verify!

need to add (full) rotation + Block size

---

**To Modify a Block?**

**To Modify Block:**

(a) Read Block  
(b) Modify in Memory  
(c) Write Block  
[(d) Verify?]

---

**An Example**  
Megatron 747 Disk  
(Example 2.1 in book)

- 4 platters, 8 surfaces  
- $2^{13} = 8192$ cylinders (or tracks/surface)  
- $2^8 = 256$ sectors/track  
  - 90% of track length for data, 10% for gaps  
- $2^9 = 512$ bytes/sector  
→ capacity $2^4 \times 2^{13} \times 2^8 \times 2^9 = 2^{33} = 8$ GB

---

**Timing for Megatron 747**  
(Ex 2.3)

- Time to read a 4 KB block?
  - **MIN** only **transfer time:** Read 8 sectors (a’ 0.5 KB) and pass 7 gaps  
  - How much does the disk need to rotate?  
    For data $8/256 \times 90% = 72/2560$ rotation,  
    for gaps $7/256 \times 10% = 7/2560$ rotation  
  - 3840 RPM $\rightarrow$ 1/64 s for one rotation  
    $\rightarrow$ time $(79/2560)/64$ s, or about 0.5 ms  
    (i.e. transfer speed approx. 8 MB/s)

---

**Maximum block access time**

- Megatron 474 head movements:  
  - 1 ms to start&stop + 1 ms/500 cylinders  

- **MAX** block access time:
  - $\text{SEEKTIME}(1 \rightarrow 8192) =$  
    $1 + 8191/500 \text{ ms} = 17.4 \text{ ms}$  
  - full rotation: $1/64 \text{ s} = 15.6 \text{ ms}$  
  - transfer: $0.5 \text{ ms}$  
    $\rightarrow$ 33.5 ms
Megatron 474: Average block access time

- **AVG** block access time:
  - AVG SEEKTIME: Assuming random access, avg head movement is 1/3 of all tracks
    → 1 ms + 1/3 x 8191/500 ms = 6.5 ms
  - half rotation: 1/2 x 1/64 s = 7.8 ms
  - transfer: 0.5 ms
  → 14.8 ms

Outline

- Hardware: Disks
- Access Times
- Example: Megatron 747
- External sorting
- Other Topics
  - Optimizing Disk Usage

Two-Phase Multiway Merge-Sort

- Sorting of files or relations is a recurrent task
- What if the data doesn’t fit in main memory?
- **Ex:** Sort relation R of 10^6 100-byte tuples, using Megatron 747 disk and 50 MB main memory (M) for buffers (ex. 2.5, 2.7-2.9, 2.14 in Book).
  - Block size 4 KB → 40 tuples/block
  → R takes 10^6/40 = 250,000 blocks
  → M can hold 50 x 2^{20/2} = 12,800 blocks MB

Merging Sorted Sublists is Easy

Always move the smallest head element to the output buffer

One way to sort: Merge Sort

(i) For each main-memory-sized chunk of R:
- Read chunk
- Sort in memory (say, using quicksort)
- Write to disk

(ii) Read all chunks + merge + write out

Sorted file

Memory

Sorted
sublists

Sorted file

Memory

Sorted
sublists
Example: Sorting R (cont.)

- **R fills memory** $250,000 / 12,800 = 20$ times (last time memory not full)
- Phase (i): 250,000 blocks first read, then written → $500,000$ block accesses. If in random order, time $500,000 \times 15$ ms = 125 minutes
- Phase (ii): Similarly 125 min; **Total 250 min**

How Large Files Can We Sort?

- Let $B = \text{block size}, M = \text{size of main memory available for sorting}, R = \text{size of each record}$
- Phase (i): $250,000$ blocks fit in memory; one for output → Phase (ii) can merge at most $(M/B - 1)$ sublists, each created by sorting at most $M/R$ records
- $M/R \times (M/B - 1)$, roughly $M^2 / (B \times R)$ records

$M^2 / (B \times R)$ in Practice?

- In our example, $M = 50,000,000, B = 4096, R = 100$
  → $M^2 / (B \times R) = 2500 \times 10^{12} / (4096 \times 100) = 6.1 \times 10^9$ records, taking $0.61$ terabytes of storage
- Pretty much!
  - Additional merge-phases can be added to cover even larger data sets.

Optimizations

- Arranging sequentially accessed blocks by cylinders
- Pre-fetching buffers
- Mirrored Disks

Arranging Blocks by Cylinders

- Consider our merge-sort example. The blocks of R can be stored on consecutive cylinders, and the 20 sorted sublists can be written likewise
  - Megatron 747 cylinder size 1 MB
  - Time to fill main memory (by 50 cylinders):
    - AVG seek 6.5 ms + 49 one-cylinder seeks $a' 1$ ms + transfer time for 12,800 blocks $a' 0.5$ ms
  - Transfer time to fill memory once 6.4 s; Seek times ignorable (if no other processes!)

Arranging Blocks by Cylinders (cont.)

- Phase (i) of multiway merge-sort dominated by the transfer time of reading & writing a memory-full of blocks 20 times:
  - $20 \times 2 \times 6.4$ s = 256 s = 4.3 min
  - (vs. previous 125 minutes!)
What about Phase (ii)?

- In the merge phase, a new block is read when one gets exhausted → blocks read in a somewhat random order
- But we wrote each sublist sequentially along cylinders!
  - We can utilize this by using larger buffers holding entire tracks or even cylinders
  - To increase throughput, apply double buffering

**Double Buffering**

Problem: Have a File
  » Sequence of Blocks B1, B2, B3, ...
  » Have a Program
  » Process B1
  » Process B2
  » Process B3

**Single Buffer Solution**

1. Read B1 → Buffer
2. Process Data in Buffer
3. Read B2 → Buffer
4. Process Data in Buffer ...

Say $P = \text{time to process/block}$
$R = \text{time to read in 1 block}$
$n = \# \text{blocks}$

Single buffer time = $n(P+R)$

Say $R \geq P$

What is processing time?

- Double buffering time = $nR$
- Single buffering time = $n(R+P)$
Cylinder-Sized Double-Buffering

- Consider implementing the merge-phase using two cylinder-sized (1 MB) buffers for each of our 20 sorted sublists
- Only one seek/cylinder (AVG 6.5 ms) + one full rotation for each of the 8 tracks of a cylinder (each 15.6 ms) → 131.3 ms to read one buffer
  - repeated 1000 times to read entire relation → about 2.15 min to read all sublists

Cylinder-Sized Double-Buffering (cont.)

- Writing the sorted relation can be arranged similarly → Time to output the buffers for the result also 2.15 min
  → Total time for multiway merge-sort 4.3 min (Phase i) + 4.3 min (Phase ii) = 8.6 minutes
  (vs. over 4 hours without these tricks!)

Mirroring Disks

- Two or more disks holding identical copies of data
  - Optimizes: Can read blocks from whichever disk is faster. (Writing on each disk happens in parallel.)
  - Enhances reliability: A single disk crash does not cause loss of data. (RAID level 1)

Using redundant disks

- RAID (Redundant Arrays of Independent Disks)
  - different versions, levels 1 - 6
  - Basic idea: One of the disks is redundant holding checksums for other disks
  - contents of crashed disks can be recovered from the checksums

Error Recovery in Databases

- Log

Current DB       Last week's DB

Summary

- Secondary storage, mainly disks
- I/O times dominate DBMS processing
- I/Os should be avoided, especially random ones

- More about this later