DATA STORAGE

(Lecture based on [GUW 11.2−11.5], [Sanders03, 1.3−1.5],
and [MaheshwariZeh03, 3.1−3.2])

Slides based on
Notes 02: Hardware
for Stanford CS 245, fall 2002
by Hector Garcia–Molina
Today

- Hardware, external memory
- Disk access times
- The I/O model
- Case study: Sorting
- Lower bound for sorting
- Optimizing disk usage
A Typical Computer

P = Processor
C = Cache
M = Main Memory

Secondary Storage (Hard Disks)
Storage Capacity

- Cache
- Electronic main
- Electronic secondary
- Optical disks
- Magnetic disks
- Online tape
- Nearline tape & optical disks
- Offline tape

Typical capacity (bytes) vs. access time (sec)

Data from Gray & Reuter
Storage Cost

from Gray & Reuter

<table>
<thead>
<tr>
<th>Storage Type</th>
<th>dollars/MB</th>
</tr>
</thead>
<tbody>
<tr>
<td>cache</td>
<td>10^4</td>
</tr>
<tr>
<td>electronic main</td>
<td>10^2</td>
</tr>
<tr>
<td>electronic secondary</td>
<td>10^0</td>
</tr>
<tr>
<td>magnetic optical disks</td>
<td>10^-2</td>
</tr>
<tr>
<td>nearline tape &amp; optical disks</td>
<td>10^-4</td>
</tr>
<tr>
<td>offline tape</td>
<td>10^-6</td>
</tr>
</tbody>
</table>

access time (sec)

[Graph showing the relationship between storage cost and access time for various types of storage media, including cache, electronic main, electronic secondary, magnetic optical disks, nearline tape & optical disks, and offline tape.]
**Caching**

**Cache:** Fast memory, holding frequently used parts of a slower, larger memory.

**Examples:**
- A small (L1) cache holds a few kilobytes of the memory "most recently used" by the processor.
- Most operating systems keep the most recently used "pages" of memory in main memory and put the rest on disk.
Virtual memory

• In most operating systems, programs don’t know if they access main memory or a page that resides on secondary memory.
• This is called **virtual memory** (the book is a little fuzzy on this).

• Database systems usually take explicit control over secondary memory accesses.
Secondary storage

Many flavors:

- Disk: Floppy (hard, soft)
  Winchester
  Ram disks
  Optical, CD–ROM…
  Arrays

- Tape Reel, cartridge
  Robots
Focus on: “Typical Disk”

Terms: – Platter, Head, Cylinder, Track Sector (all physical).
– Block (logical).
“Typical” Numbers

Diameter: 1 inch → 15 inches

Cylinders: 40 (floppy) → 20000

Surfaces: 1 (CDs) →
          2 (floppies) → 30

Sector Size: 512B → 50K

Capacity: 360 KB (old floppy)
          → 30 GB (I use)
Disk Access Time

I want block X

?}

block x in memory
Time = Seek Time + Rotational Delay + Transfer Time + Other
Seek Time

![Graph showing seek time as a function of cylinders traveled. The graph indicates that the seek time increases significantly as the number of cylinders increases, reaching multiples of x or 5x.]
Average Random Seek Time

\[ S = \sum_{i=1}^{N} \sum_{j=1, j \neq i}^{N} \text{SEEKTIME (i \rightarrow j)} \]

\[ S = \frac{N(N-1)}{N(N-1)} \]

“Typical” \( S \): 10 ms \( \rightarrow \) 40 ms

10s of millions clock cycles!
Rotational Delay

Head Here

Block I Want
Average Rotational Delay

\[ R = \frac{1}{2} \text{ revolution} \]

“typical” \[ R = 4.16 \text{ ms} \ (7200 \text{ RPM}) \]
Transfer Rate: $t$

- “typical” $t$: $10 \rightarrow 50$ MB/second
- transfer time: $\frac{\text{block size}}{t}$
- E.g., block size 32 kb, $t=32$ MB/second gives transfer time 1 ms.
Other Delays

- CPU time to issue I/O
- Contention for controller
- Contention for bus, memory

“Typical” Value: 0
• **So far:** Random Block Access
• **What about:** Reading “Next” block?
If we don’t need to change cylinder

Time to get = Block Size + Negligible

block t

– switch track
– once in a while,
   next cylinder
Rule of Thumb

Random I/O: Expensive
Sequential I/O: Less expensive

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

However, the relative difference is smaller if we use larger blocks.
Cost for Writing similar to Reading

.... unless we want to verify!

need to add (full) rotation + Block size
To Modify a Block?

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

We usually express the running time of algorithms as the number of operations.  
• Give an example of algorithms (using external memory) where this is misleading.  
• What should we count instead?  
• How do we do the counting?
The I/O model of computation

• Count the number of disk blocks read or written by an algorithm (I/Os).
• Ignore the number of operations! (?)
• Explicit control of which blocks are in main memory.

• **Notation:** \( M = \text{size of main memory} \)
  \( B = \text{size of disk blocks} \)
  \( N = \text{size of data set} \)
Problem session

Name a few sorting algorithms that you know:

• What are the (worst case) running times in internal memory?
• What is the number of I/Os if we use the algorithm on external memory
  – if no caching is done?
  – when storing the M/B most recently accessed blocks in main memory?
External memory sorting

Let’s see if we can do better..!

(Perhaps as well as in [MaheshwariZeh03, 3.2])
Sorting in GUW

- More practical viewpoint: Two passes over the data enough unless data is huge.
- **TPMMS** = Two–Pass Multiway MergeSort.
- More general treatment in [MaheshwariZeh03, 3.2]
External memory sorting

Let’s see if we did as well as we possibly could:

A lower bound on the number of I/Os for sorting

(based on [Sanders03, 1.5])
Problem session

Consider the sorting problem in example 11.7 (page 527 in GUW):

- What are the values of N, B, and M?
- How many I/Os will our sorting algorithm use on this problem?
- What lower bound does the formula from [Sanders03, Theorem 1] give?
**Optimizations**

Some practically oriented ways of speeding up external memory algorithms:

- **Disk Scheduling**
  - e.g., using the "elevator algorithm"
  - reduces average seek time when there are multiple simultaneous "unpredictable" requests

- **Track buffer / cylinder buffer**
  - good when data are arranged and accessed "by cylinder"
• Pre-fetching
  – Speeds up access when the needed blocks are known, but the order of requests is data dependent

• Disk arrays
  – Increases the rate at which data can be transferred
  – **Striping**: Blocks from each disk are grouped to form a large logical block

• Mirrored disks
  – Same speed for writing
  – Several blocks can be read in parallel
Block Size Selection?

- Big Block $\rightarrow$ Amortize I/O Cost

Unfortunately...

- Big Block $\Rightarrow$ Read in more useless stuff!
  And takes longer to read
- **Trend**: Blocks get bigger
Problem session

Which of the mentioned optimizations apply to (parts of) the sorting algorithm we developed earlier?

• Disk scheduling
• Cylinder buffer
• Prefetching
• Disk arrays
• Mirrored disks
Summary

• External memories are complicated.
• Essential features captured by the I/O model (to be used henceforth).
• We saw matching I/O upper and lower bounds for sorting.
• A bit (and the last bit) about optimizing constant factors in external memory access.