Outline

- Hardware: Disks
- Access Times
- Example - Megatron 747
- Optimizations
- Other Topics:
  - Storage costs
  - Using secondary storage
  - Disk failures

Hardware

DBMS

Data Storage

Typical Computer

Secondary Storage

Processor

Fast, slow, reduced instruction set, with cache, pipelined...
Speed: 100 → 500 → 1000 MIPS

Memory

Fast, slow, non-volatile, read-only,...
Access time: $10^{-6} \rightarrow 10^{-9}$ sec.
1 µs → 1 ns

Secondary storage

Many flavors:
- Disk: Floppy (hard, soft) Removable Packs Winchester Ram disks Optical, CD-ROM...
Arrays
- Tape: Reel, cartridge Robots
Focus on: “Typical Disk”

Terms: Platter, Head, Actuator
Cylinder, Track
Sector (physical), Block (logical), Gap

“Typical” Numbers
- Diameter: 1 inch → 15 inches
- Cylinders: 100 → 2000
- Surfaces: 1 (CDs) → 2 (floppies) → 30
- Sector Size: 512B → 50K
- Capacity: 360 KB (old floppy) → 1 TB (I use)

Disk Access Time
I want block X in memory

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

Seek Time

3 or 5x

Time

x

1

N

Cylinders Traveled
Average Random Seek Time

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

“Typical” S: 10 ms → 40 ms

Average Rotational Delay

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

“typical” R = 8.33 ms (3600 RPM)

Transfer Rate: t

- “typical” t: 10’s → 100’s MB/second
- transfer time: \( \frac{\text{block size}}{t} \)

Other Delays

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

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

“Typical” Value: 0

Other Delays (now and near future)

- Increasing amount of parallelism
- Contention can become a problem
- -> need rethink approach to scale

So far: Random Block Access
What about: Reading “Next” block?

If we do things right (e.g., Double Buffer, Stagger Blocks...)

Time to get = Block Size + Negligible block
t
- 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
t
• To Modify a Block?

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

Block Address:
- Physical Device
- Cylinder #
- Surface #
- Sector

Complication: Bad Blocks
- Messy to handle
- May map via software to integer sequence
  \[ \{1, 2, \ldots, m\} \rightarrow \text{Actual Block Addresses} \]

An Example  Megatron 747 Disk (old)
- 3.5 in diameter
- 3600 RPM
- 1 surface
- 16 MB usable capacity (16 \times 2^{20})
- 128 cylinders
- seek time: average = 25 ms. adjacent cyl = 5 ms.

1 KB blocks = sectors
10% overhead between blocks
capacity = 16 MB = (2^{20})16 = 2^{24}
# cylinders = 128 = 2^7
bytes/cyl = 2^{24}/2^7 = 2^{17} = 128 KB
blocks/cyl = 128 KB / 1 KB = 128
3600 RPM $\rightarrow$ 60 revolutions / sec  
$\quad \rightarrow$ 1 rev. = 16.66 msec. 

One track: 

---

3600 RPM $\rightarrow$ 60 revolutions / sec  
$\quad \rightarrow$ 1 rev. = 16.66 msec. 

One track: 

Time over useful data: $(16.66)(0.9)=14.99$ ms.  
Time over gaps: $(16.66)(0.1) = 1.66$ ms.  
Transfer time 1 block = $14.99/128=0.117$ ms.  
Trans. time 1 block+gap = $16.66/128=0.13$ms.

---

Burst Bandwidth 

1 KB in 0.117 ms. 

$BB = \frac{1}{0.117} = 8.54$ KB/ms. 

or 

$BB = 8.54 \text{KB/ms} \times 1000 \text{ms/1sec} \times 1\text{MB/1024KB} = \frac{8540}{1024} = 8.33$ MB/sec

---

Sustained bandwith (over track) 

128 KB in 16.66 ms. 

$SB = \frac{128}{16.66} = 7.68$ KB/ms 

or 

$SB = 7.68 \times 1000/1024 = 7.50$ MB/sec.

---

$T_1 = \text{Time to read one random block}$ 

$T_1 = \text{seek} + \text{rotational delay} + \text{TT}$ 

$= 25 + \frac{16.66}{2} + .117 = 33.45$ ms.

---

Suppose OS deals with 4 KB blocks 

$T_4 = 25 + \frac{16.66}{2} + (.117) \times 1$ 

$+ (.130) \times 3 = 33.83$ ms 

[Compare to $T_1 = 33.45$ ms]
\[ T_T = \text{Time to read a full track} \]
\[ T_T = 25 + (0.130/2) + 16.66^* = 41.73 \text{ ms} \]

\[ \text{to get to first block} \]

\* Actually, a bit less; do not have to read last gap.

The **NEW** Megatron 747
- 8 Surfaces, 3.5 Inch diameter
  - outer 1 inch used
- \(2^{13} = 8192 \text{ Tracks/surface} \)
- 256 Sectors/track
- \(2^9 = 512 \text{ Bytes/sector} \)

- 8 GB Disk
- If all tracks have 256 sectors
  - Outermost density: 100,000 bits/inch
  - Inner density: 250,000 bits/inch

- Outer third of tracks: 320 sectors
- Middle third of tracks: 256
- Inner third of tracks: 192

- Density: \(114,000 \rightarrow 182,000 \text{ bits/inch} \)

Timing for **new** Megatron 747 (Ex 2.3)
- Time to read 4096-byte block:
  - MIN: 0.5 ms
  - MAX: 33.5 ms
  - AVE: 14.8 ms

Outline
- Hardware: Disks
- Access Times
- Example: Megatron 747
- Optimizations
- Other Topics
  - Storage Costs
  - Using Secondary Storage
  - Disk Failures
**Optimizations** (in controller or O.S.)

- Disk Scheduling Algorithms
  - e.g., elevator algorithm
- Track (or larger) Buffer
- Pre-fetch
- Arrays
- Mirrored Disks
- On Disk Cache

**Double Buffering**

Problem: Have a File
- Sequence of Blocks B1, B2

Have a Program
- Process B1
- Process B2
- Process B3

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

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

---

**Single Buffer Solution**

1. Read B1 $\rightarrow$ Buffer  
2. Process Data in Buffer  
3. Read B2 $\rightarrow$ Buffer  
4. Process Data in Buffer ...

---

**Double Buffering**

Memory:

```
   A   B
   C   D
```

Disk:

```
A B C D E F G
```

---

**Double Buffering**

Memory:

```
   A   B
```

Disk:

```
B C D E F G
```
Double Buffering

Memory:

Disk:

Double Buffering

Memory:

Disk:

Say \( P \geq R \)

\[ P = \text{Processing time/blk} \]

\[ R = \text{IO time/blk} \]

\[ n = \# \text{ blocks} \]

What is processing time?

\[ \text{Double buffering time} = R + nP \]

\[ \text{Single buffering time} = n(R+P) \]

Disk Arrays

- RAIDs (various flavors)
- Block Striping
- Mirrored

On Disk Cache

\[ P \]

\[ \ldots \]

\[ M \]

\[ C \]

\[ \ldots \]

cache

\[ \text{cache} \]

\[ \text{logically one disk} \]
Block Size Selection?

- Big Block → Amortize I/O Cost, Less Management Overhead

Unfortunately...

- Big Block ⇒ Read in more useless stuff! and takes longer to read

Trend

- As memory prices drop, blocks get bigger ...

Using secondary storage effectively

- Example: Sorting data on disk
- Conclusion:
  - I/O costs dominate
  - Design algorithms to reduce I/O
- Also: How big should blocks be?
Five Minute Rule

- Say a page is accessed every X seconds
- \( CD = \text{cost if we keep that page on disk} \)
  - \( D = \text{cost of disk unit} \)
  - \( I = \text{numbers IOs that unit can perform per second} \)
  - In X seconds, unit can do XI IOs
  - So \( CD = \frac{D}{XI} \)

Five Minute Rule

- Say a page is accessed every X seconds
- \( CM = \text{cost if we keep that page on RAM} \)
  - \( M = \text{cost of 1 MB of RAM} \)
  - \( P = \text{numbers of pages in 1 MB RAM} \)
  - So \( CM = \frac{M}{P} \)

Five Minute Rule

- Say a page is accessed every X seconds
- If CD is smaller than CM,
  - keep page on disk
  - else keep in memory
- Break even point when \( CD = CM \), or
  \( X = \frac{D \cdot P}{I \cdot M} \)

Using ‘97 Numbers

- \( P = 128 \text{ pages/MB (8KB pages)} \)
- \( I = 64 \text{ accesses/sec/disk} \)
- \( D = 2000 \text{ dollars/disk (9GB + controller)} \)
- \( M = 15 \text{ dollars/MB of DRAM} \)
- \( X = 266 \text{ seconds (about 5 minutes)} \)
  (did not change much from 85 to 97)

Disk Failures

- Partial \( \rightarrow \) Total
- Intermittent \( \rightarrow \) Permanent

Coping with Disk Failures

- Detection
  - e.g. Checksum
- Correction
  \( \Rightarrow \) Redundancy
At what level do we cope?

- Single Disk
  - e.g., Error Correcting Codes
- Disk Array

Operating System

- e.g., Stable Storage

  Logical Block  →  Copy A  →  Copy B

Logical  →  Physical

Database System

- e.g.,

  Current DB  →  Last week’s DB

Log

Summary

- Secondary storage, mainly disks
- I/O times + formulas
  - Sequential vs. random
- I/Os should be avoided, especially random ones.....
- OS optimizations
- Disk errors

Outline

- Hardware: Disks
- Access Times
- Example: Megatron 747
- Optimizations
- Other Topics
  - Storage Costs
  - Using Secondary Storage
  - Disk Failures
  - Here

Outlook - Hardware

- Disk Access is the main limiting factor
- However, to implement fast DBMS
  - need to understand other parts of the hardware
    - Memory hierarchy
    - CPU architecture: pipelining, vector instructions, OOE, ...
    - SSD storage
  - need to understand how OS manages hardware
    - File access, VM, Buffering, ...
Memory Hierarchy

- **CPU Register** (< 1KB, 1 cycle)
- **L1 Cache** (10 KB, few cycles)
- **L2 Cache** (e.g., 512 KB, 2-10 x L1)
- **L3 Cache** (MB)
- **Main Memory** (GB, 100’s cycles)

**Compare:** Disk vs. Main Memory
- Reduce accesses to main memory
- Cache conscious algorithms

Increasing Amount of Parallelism
- Contention on, e.g., Memory
- NUMA
- Algorithmic Challenges
  - How to parallelize algorithms?
  - Sometime: Completely different approach required
  - -> Rewrite large parts of DBMS

New Trend: Software/Hardware Co-design
- Actually, revived trend: database machines (80’s)
- New goals: power consumption
- Design specific hardware and write special software for it
- E.g., Oracle Exadata, Oracle Labs