CS 525: Advanced Database Organization

02: Hardware

Boris Glavic

Slides: adapted from a course taught by Hector Garcia-Molina, Stanford InfoLab

COMPUTER SCIENCE

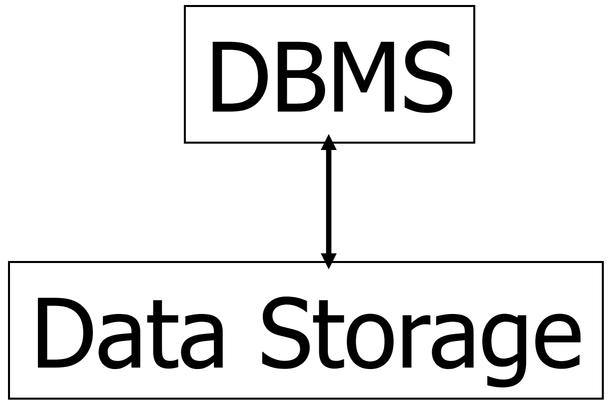
IIT College of Science and Letters

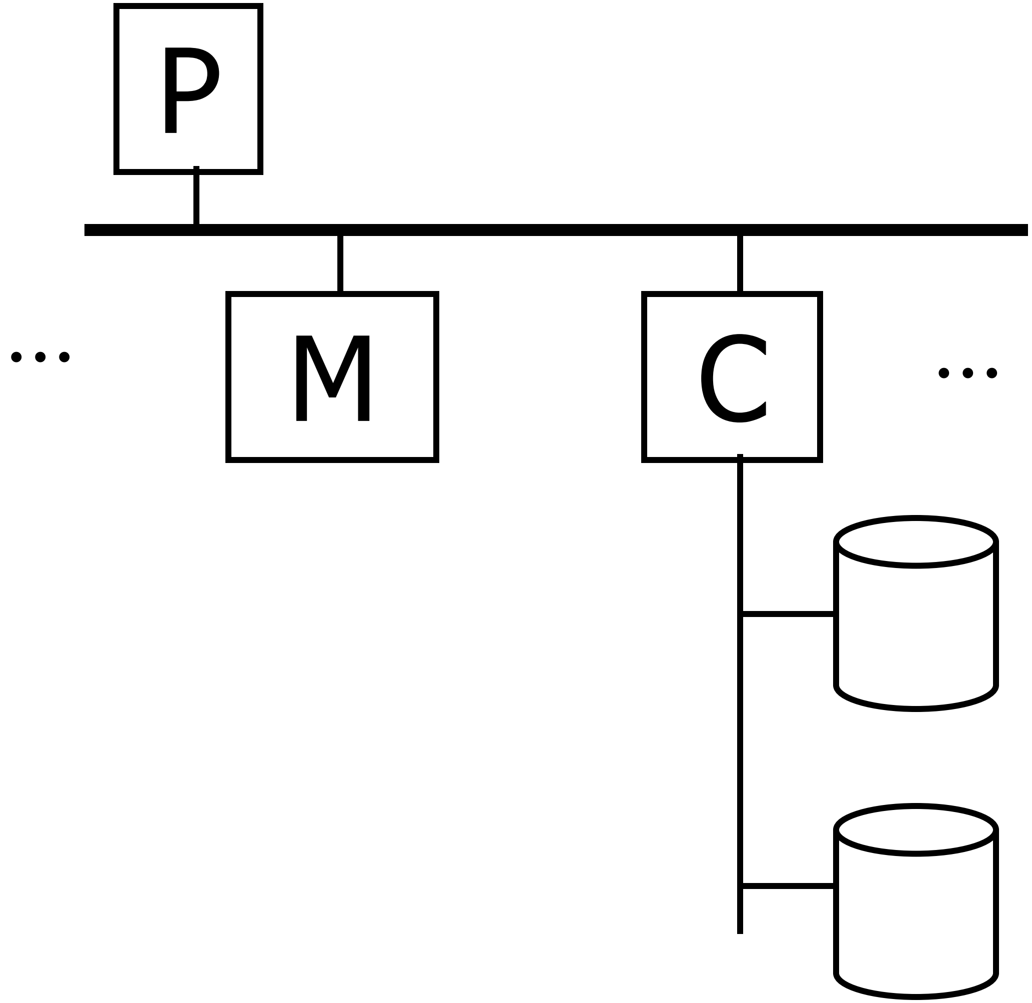
ILLINOIS INSTITUTE OF TECHNOLOGY

Outline

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

Hardware





<u>Typical Computer</u>

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} → 10^{-9} sec.

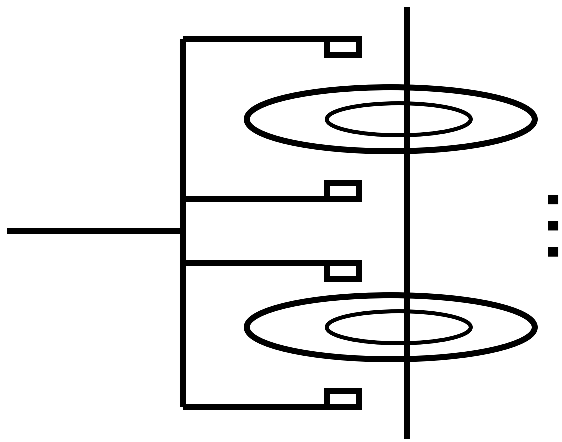
1 μs → 1 ns

Secondary storage

Many flavors:

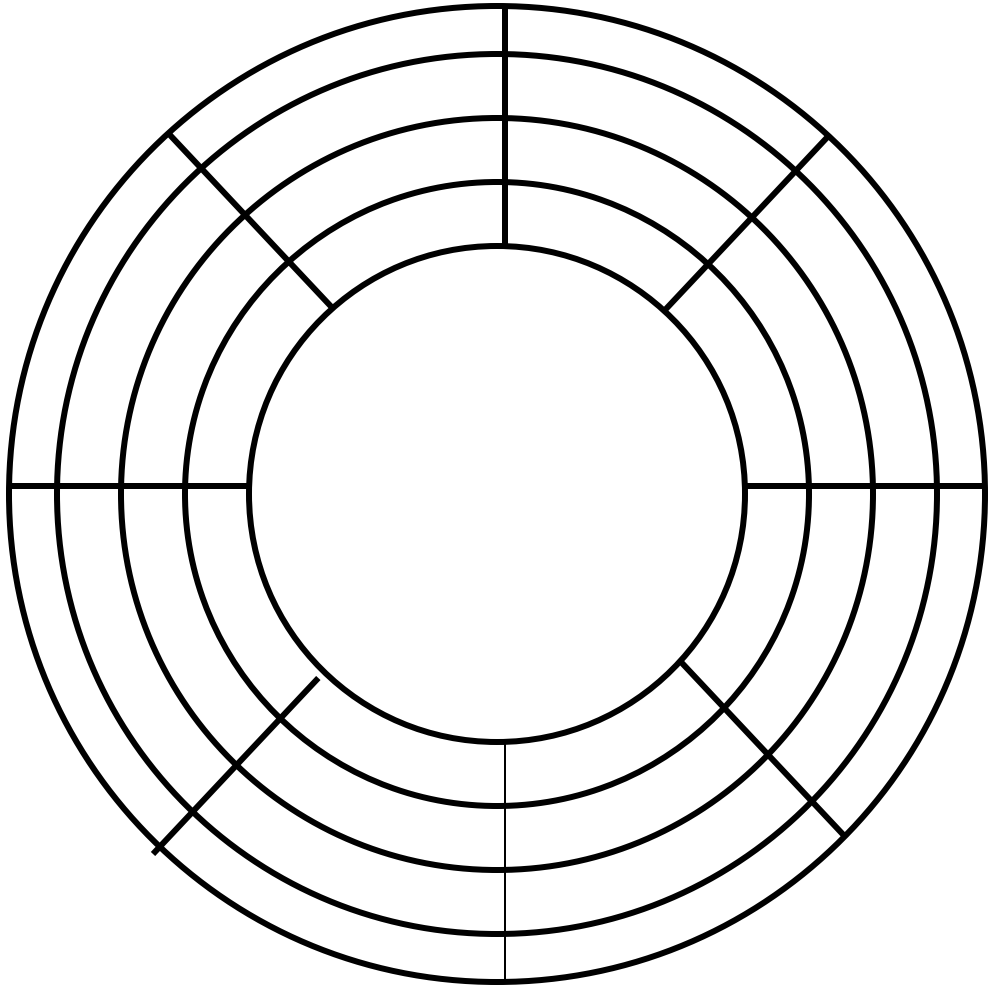
- Disk: Floppy (hard, soft)
  Removable Packs
  Winchester
  Ram disks
  Optical, CD-ROM…
  Arrays
- Tape Reel, cartridge
  Robots

Focus on: "Typical Disk"



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

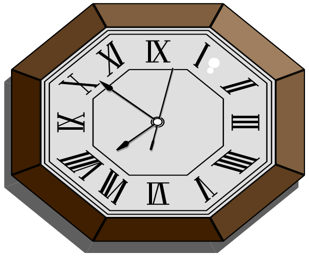
Top View



“Typical” Numbers

Diameter:	1 inch → 15 inches
Cylinders:	100 → 2000
Surfaces:	1 (CDs) →
(Tracks/cyl)	2 (floppies) → 30
Sector Size:	512B → 50K
Capacity:	360 KB (old floppy) → 1 TB (I use)

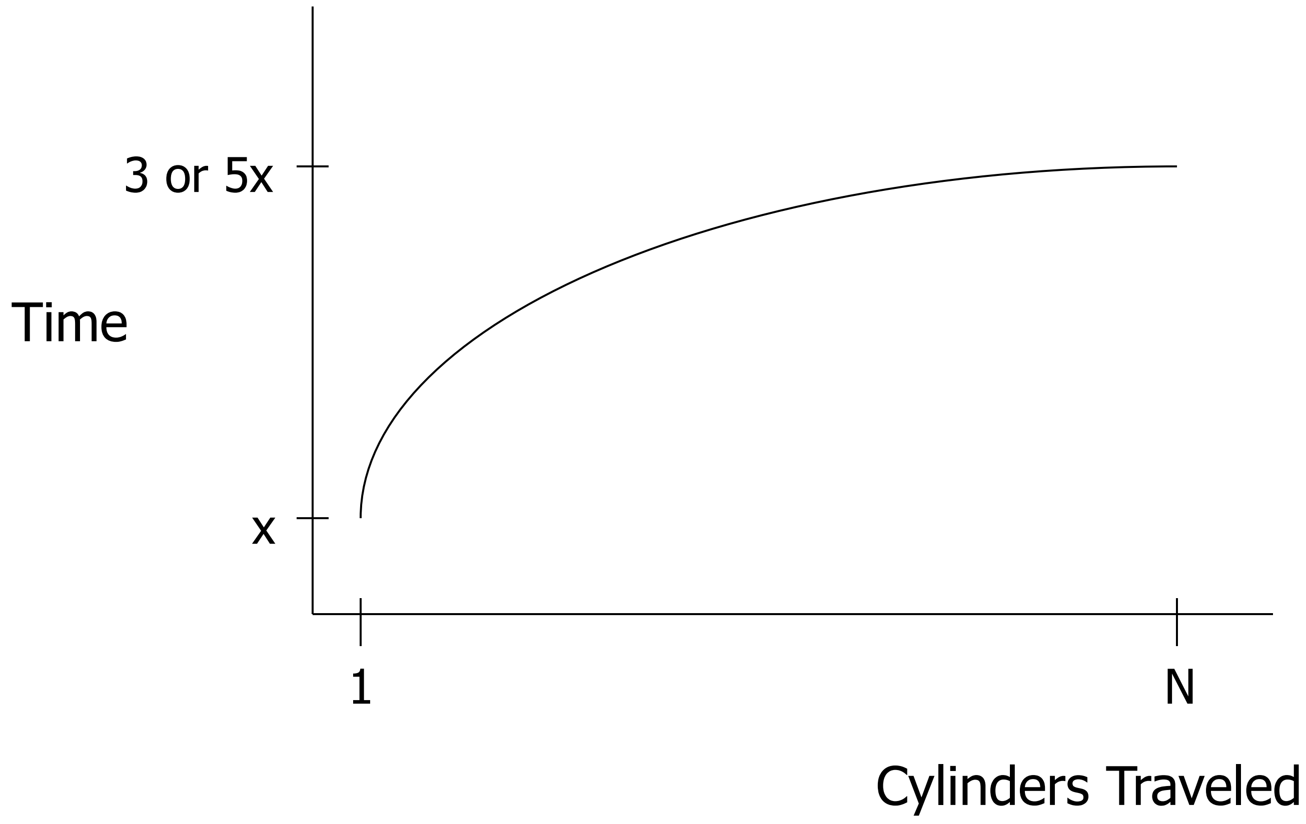
Disk Access Time

I want block X →  → block x in memory

?

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

Seek Time



Average Random Seek Time

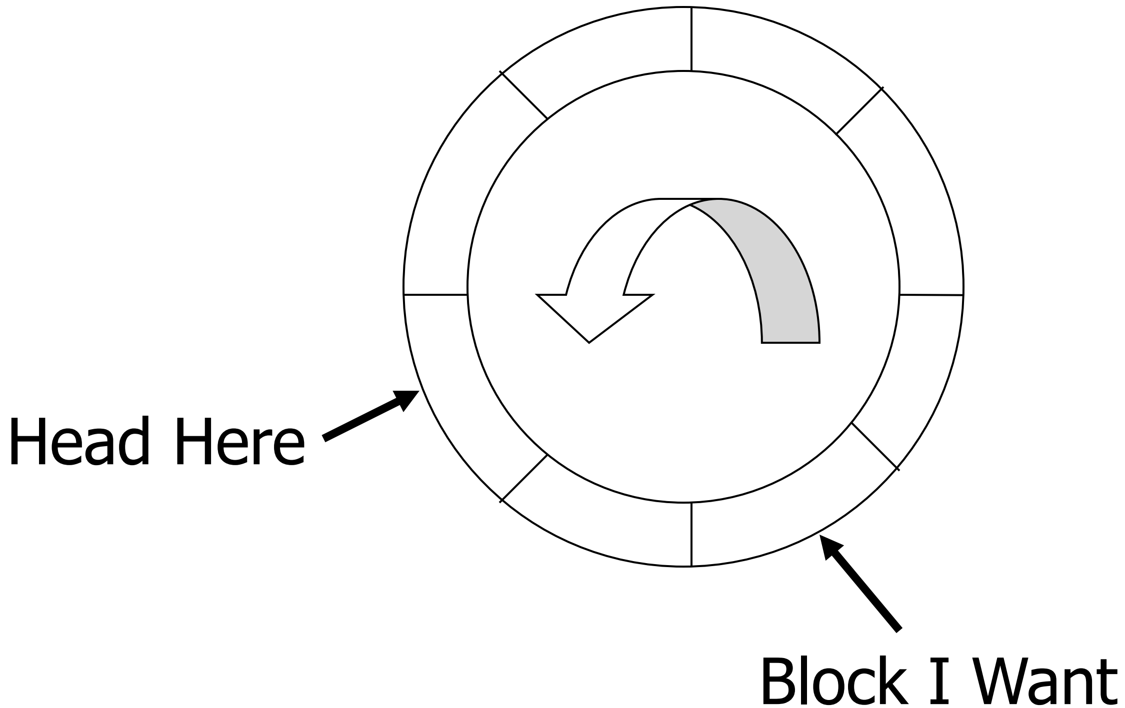
$$S = \frac{\displaystyle\sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} \text{SEEKTIME}(i \rightarrow j)}{N(N-1)}$$

Average Random Seek Time

$$S = \frac{\sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} \text{SEEKTIME } (i \rightarrow j)}{N(N-1)}$$

"Typical" S: 10 ms → 40 ms

Rotational Delay



Average Rotational Delay

R = 1/2 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?

<u>If we do things right</u> (e.g., Double Buffer, Stagger Blocks…)

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

- 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 + $\frac{\text{Block size}}{t}$

- To Modify a Block?

- To Modify a Block?

To Modify Block:

(a) Read Block

(b) Modify in Memory

(c) Write Block

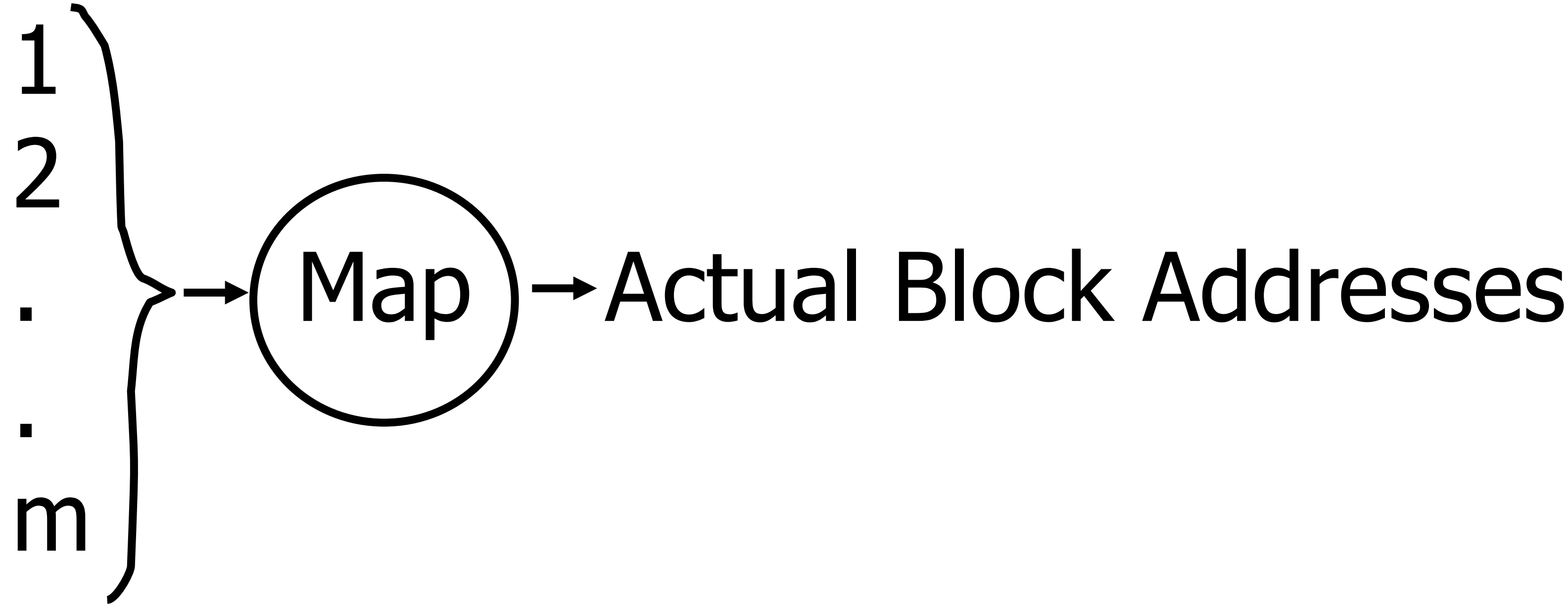
[(d) Verify?]

Block Address:

- Physical Device
- Cylinder #
- Surface #
- Sector

Complication: Bad Blocks

- Messy to handle
- May map via software to integer sequence



An Example Megatron 747 Disk (old)

- 3.5 in diameter
- 3600 RPM
- 1 surface
- 16 MB usable capacity (16×2^{20})
- 128 cylinders
- seek time: average = 25 ms.

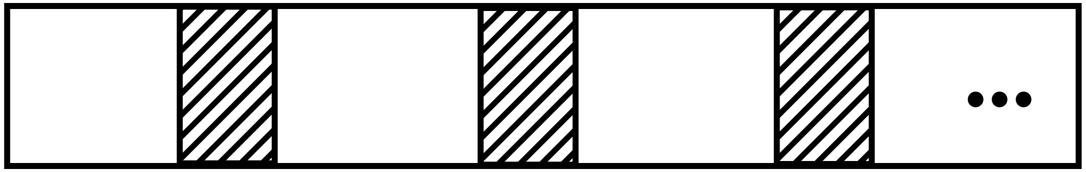
  adjacent cyl = 5 ms.

- 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 → 60 revolutions / sec

→ 1 rev. = 16.66 msec.

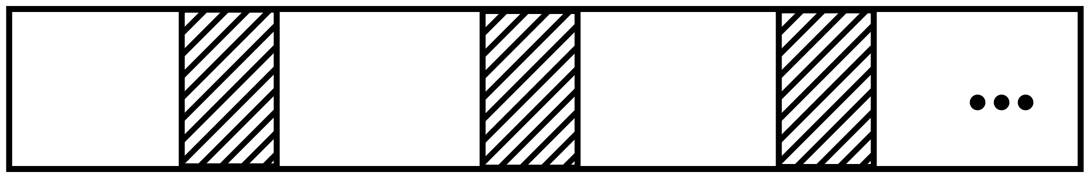
One track:



3600 RPM → 60 revolutions / sec

→ 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=0.13ms.

Burst Bandwith

1 KB in 0.117 ms.

BB = 1/0.117 = 8.54 KB/ms.

or

BB =8.54KB/ms x 1000 ms/1sec x 1MB/1024KB
= 8540/1024 = 8.33 MB/sec

Sustained bandwith (over track)

128 KB in 16.66 ms.

SB = 128/16.66 = 7.68 KB/ms

or

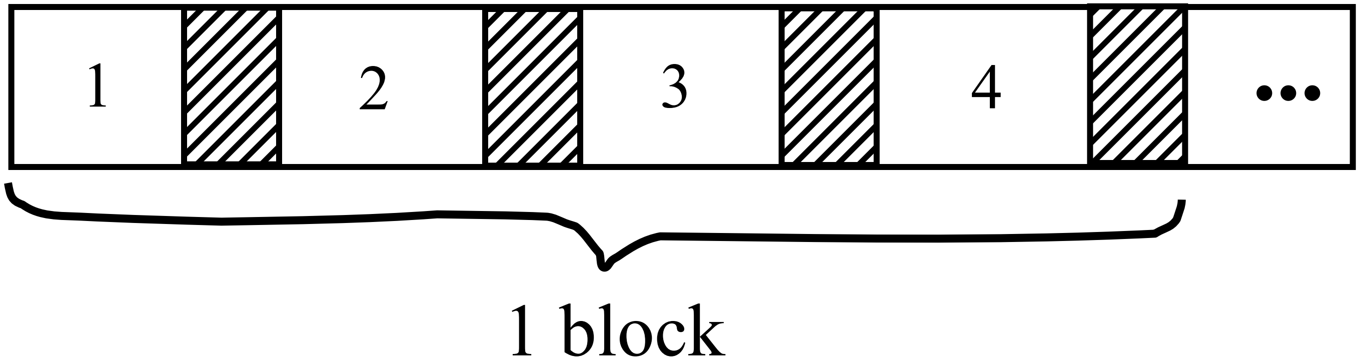
SB = 7.68 x 1000/1024 = 7.50 MB/sec.

T_1 = Time to read one random block

T_1 = seek + rotational delay + TT

= 25 + (16.66/2) + .117 = 33.45 ms.

Suppose OS deals with 4 KB blocks



$T_4 = 25 + (16.66/2) + (.117) \times 1$
$+ (.130) \times 3 = 33.83$ ms

[Compare to $T_1 = 33.45$ ms]

T_T = Time to read a full track
(start at any block)

$T_T = 25 + (0.130/2) + 16.66^{*} = 41.73$ ms

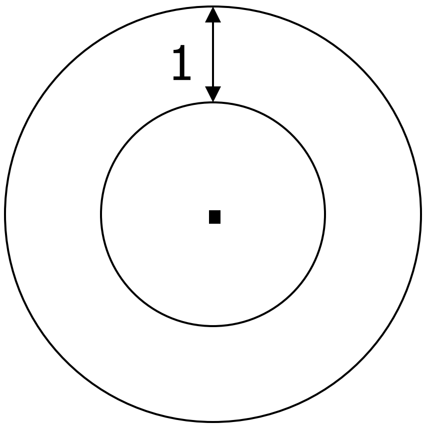
to get to first block

* 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 Tracks/surface
- 256 Sectors/track
- 2^{9} = 512 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 → 182,000 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

here

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

⋮

Single Buffer Solution

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

Say P = time to process/block

R = time to read in 1 block

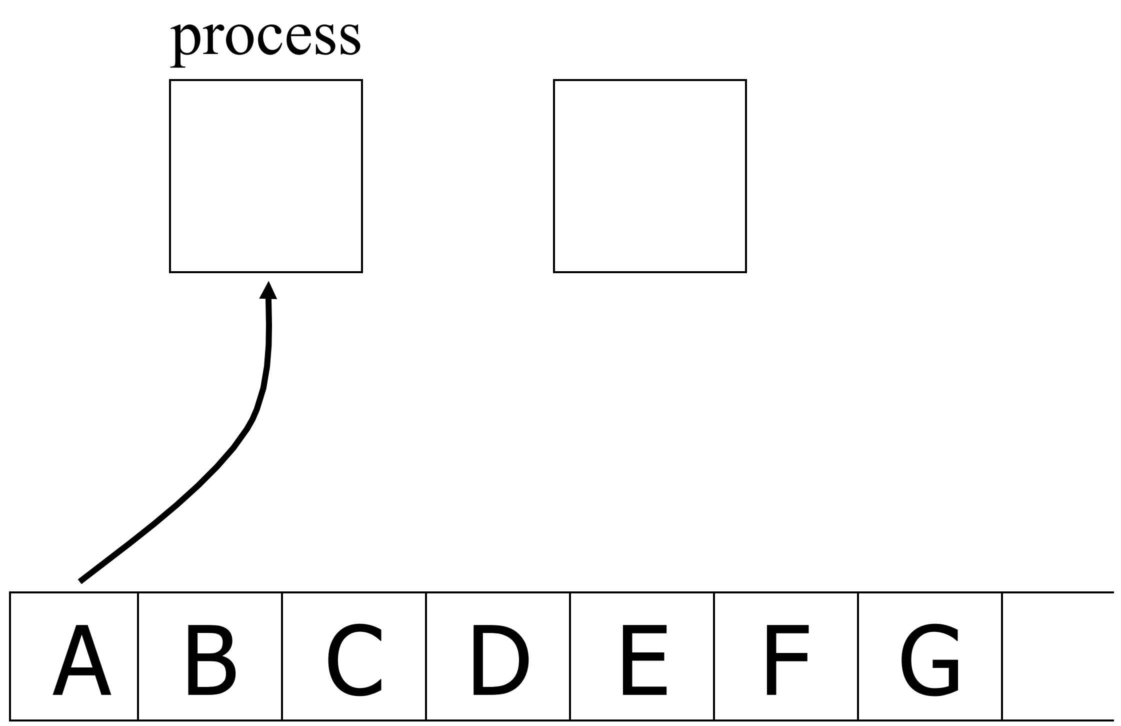
n = # blocks

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

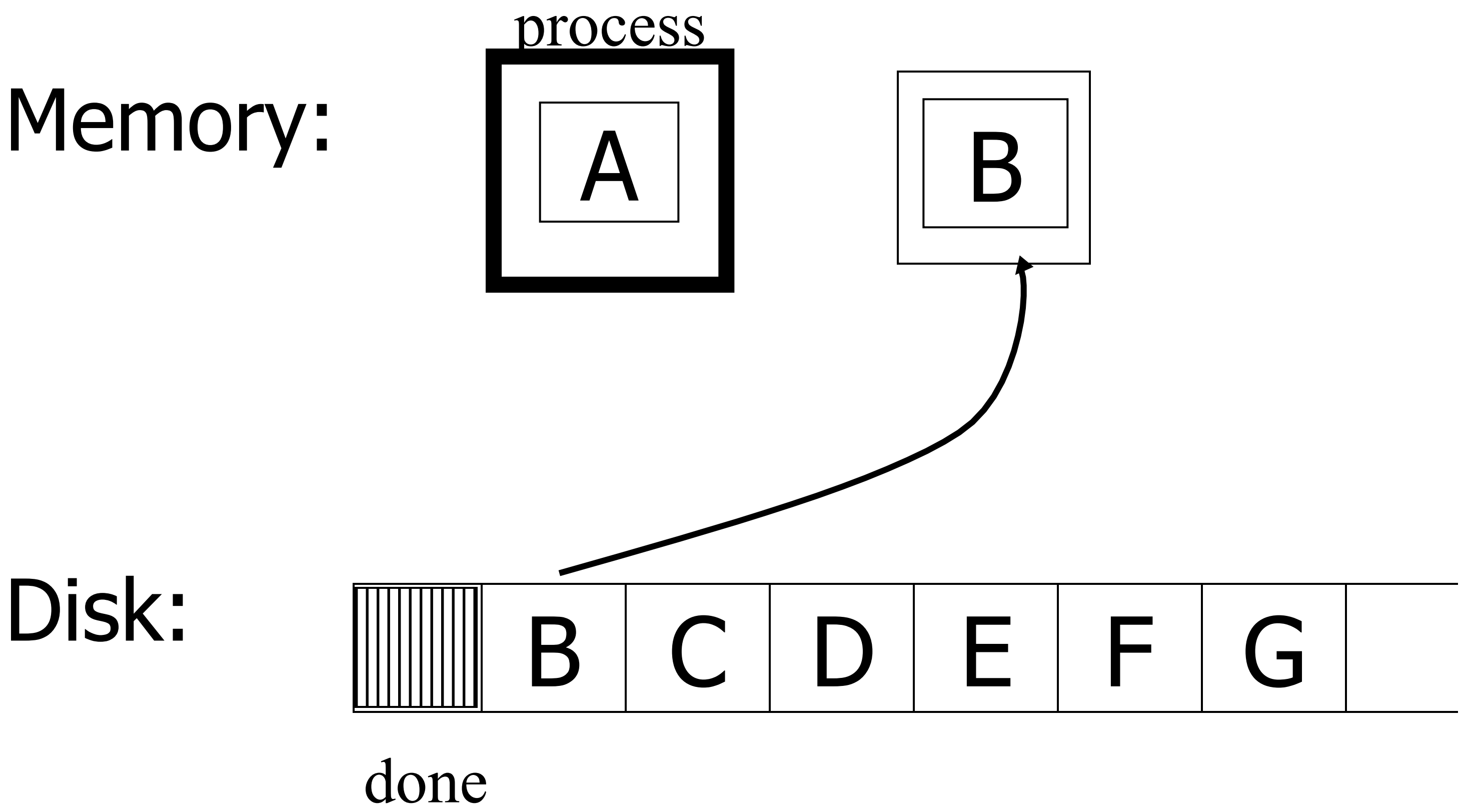
Double Buffering

Memory:

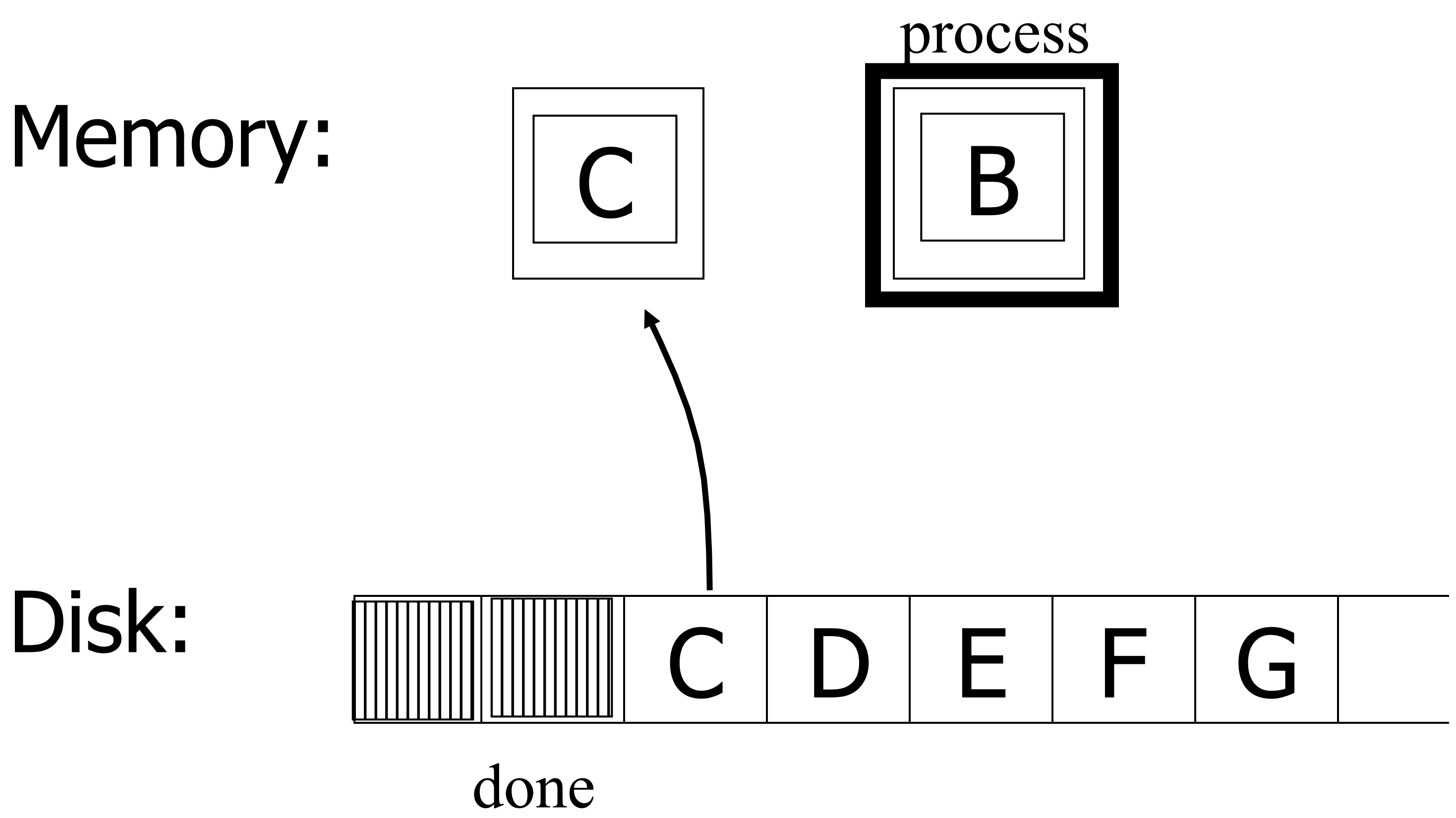
Disk:



Double Buffering



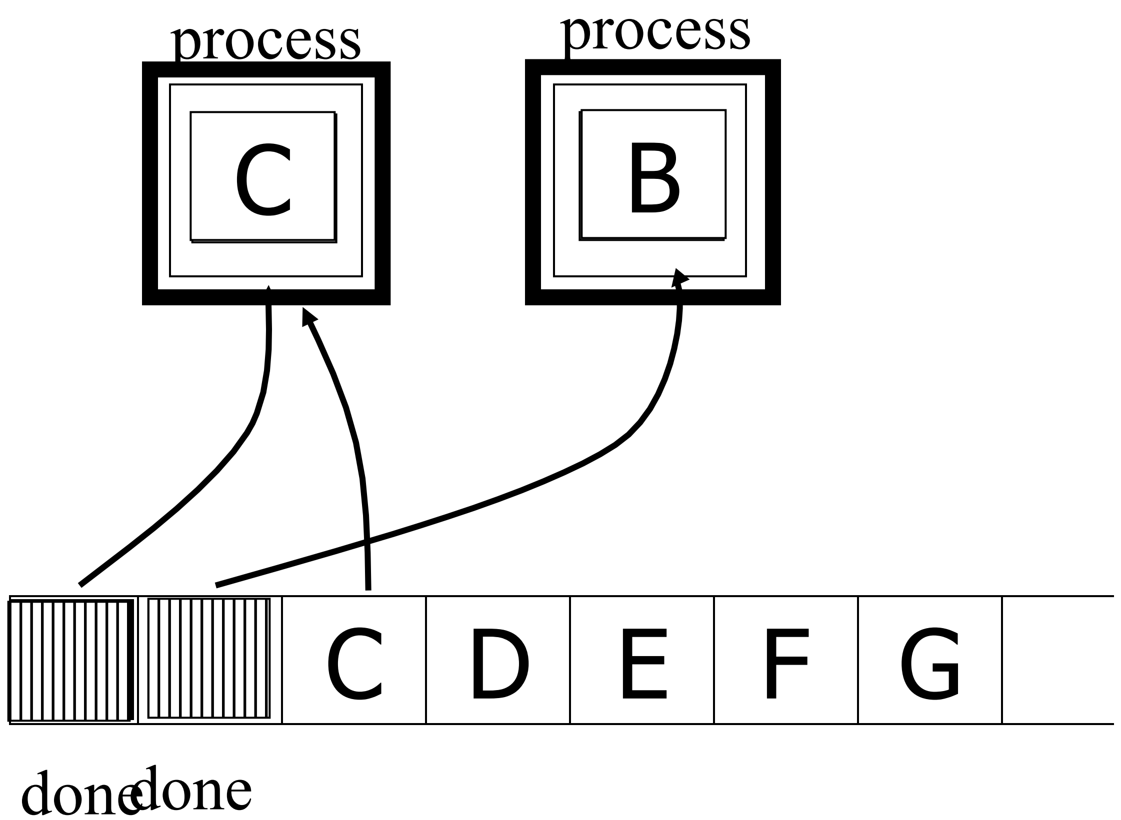
Double Buffering



Double Buffering

Memory:

Disk:



Say $P \geq R$

P = Processing time/block
R = IO time/block
n = # blocks

What is processing time?

Say $P \geq R$

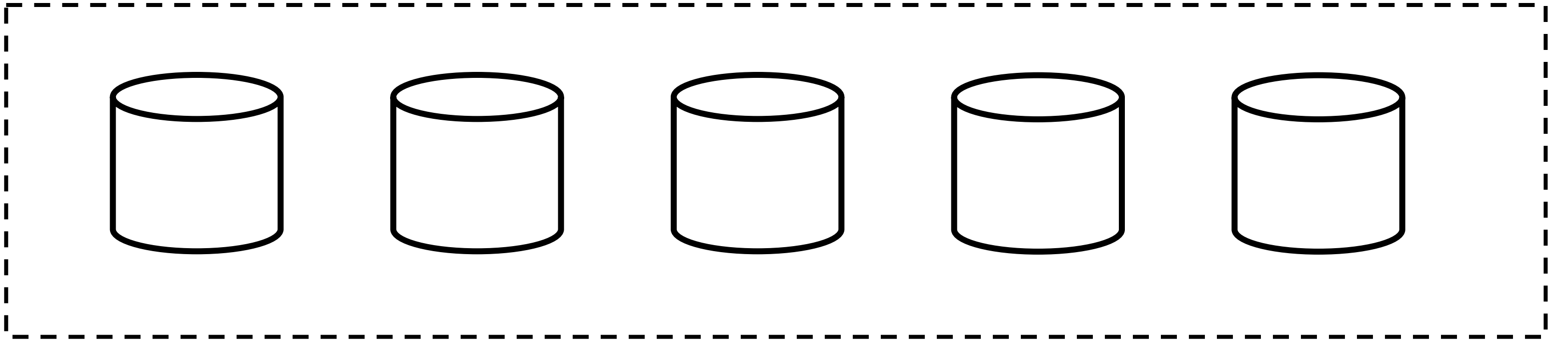
P = Processing time/block
R = IO time/block
n = # blocks

What is processing time?

- Double buffering time = $R + nP$
- Single buffering time = $n(R+P)$

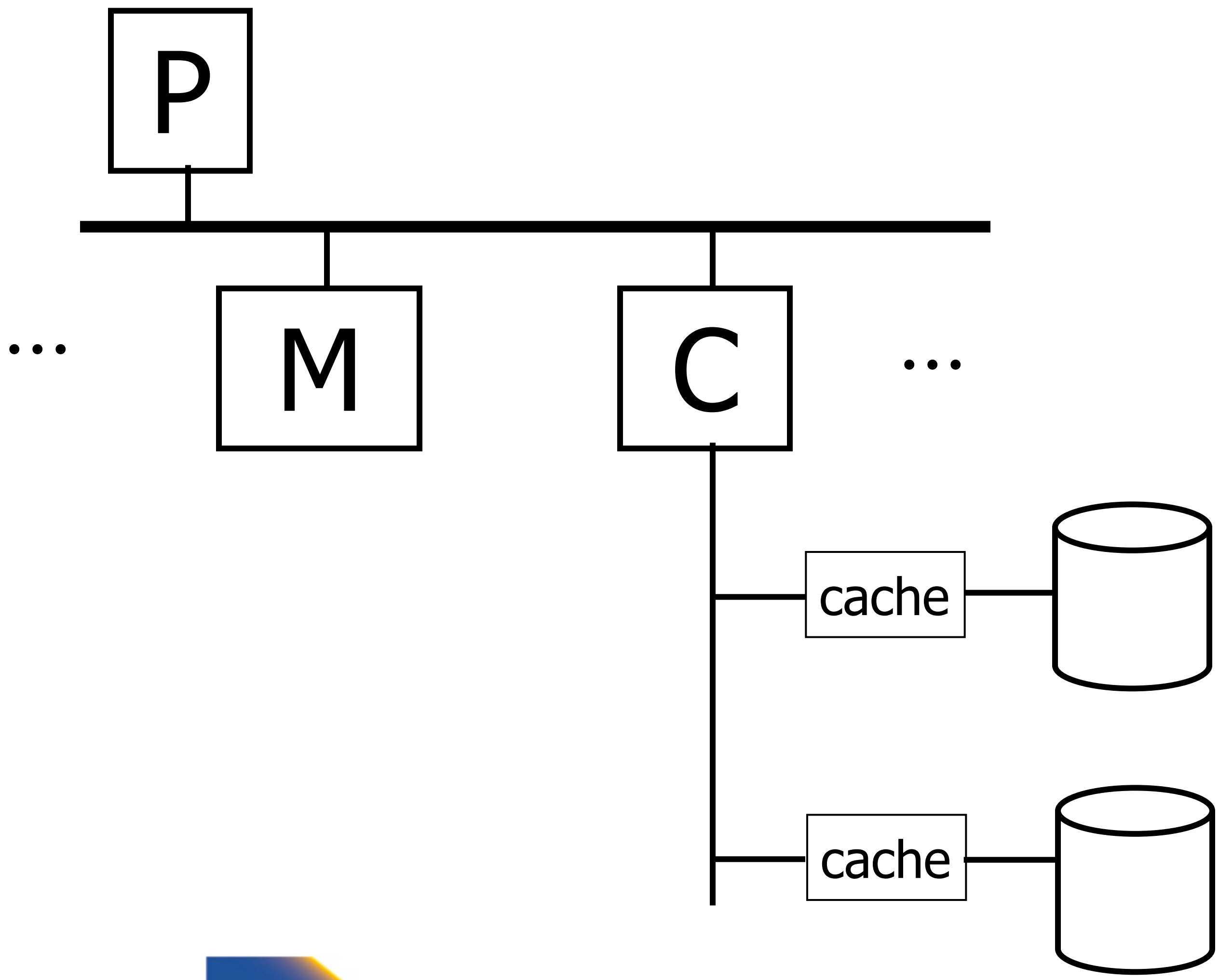
Disk Arrays

- RAIDs (various flavors)
- Block Striping
- Mirrored



logically one disk

On Disk Cache



Block Size Selection?

- Big Block $\rightarrow$ Amortize I/O Cost, Less Management Overhead

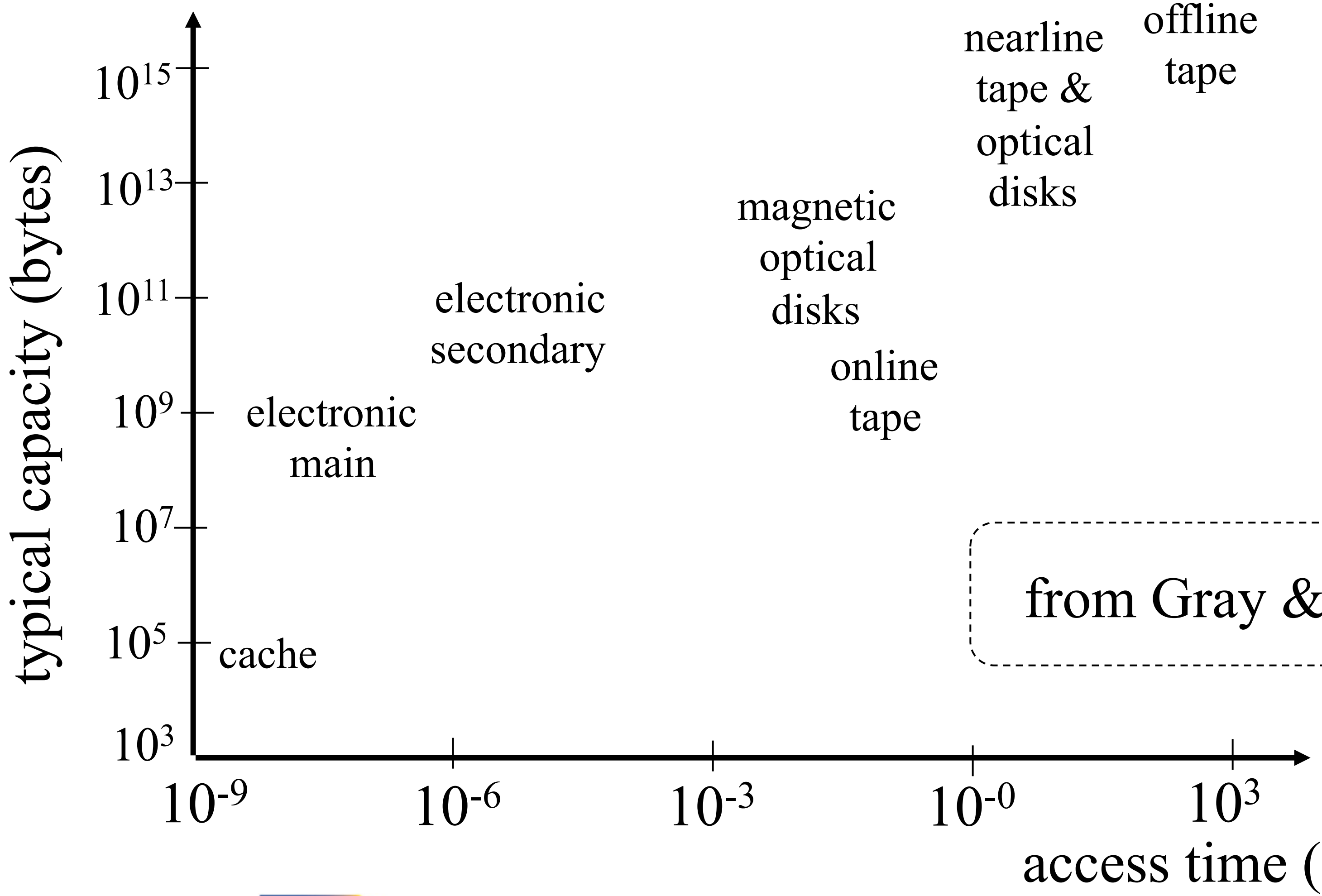
Unfortunately...

- Big Block $\Rightarrow$ Read in more useless stuff! and takes longer to read

Trend

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

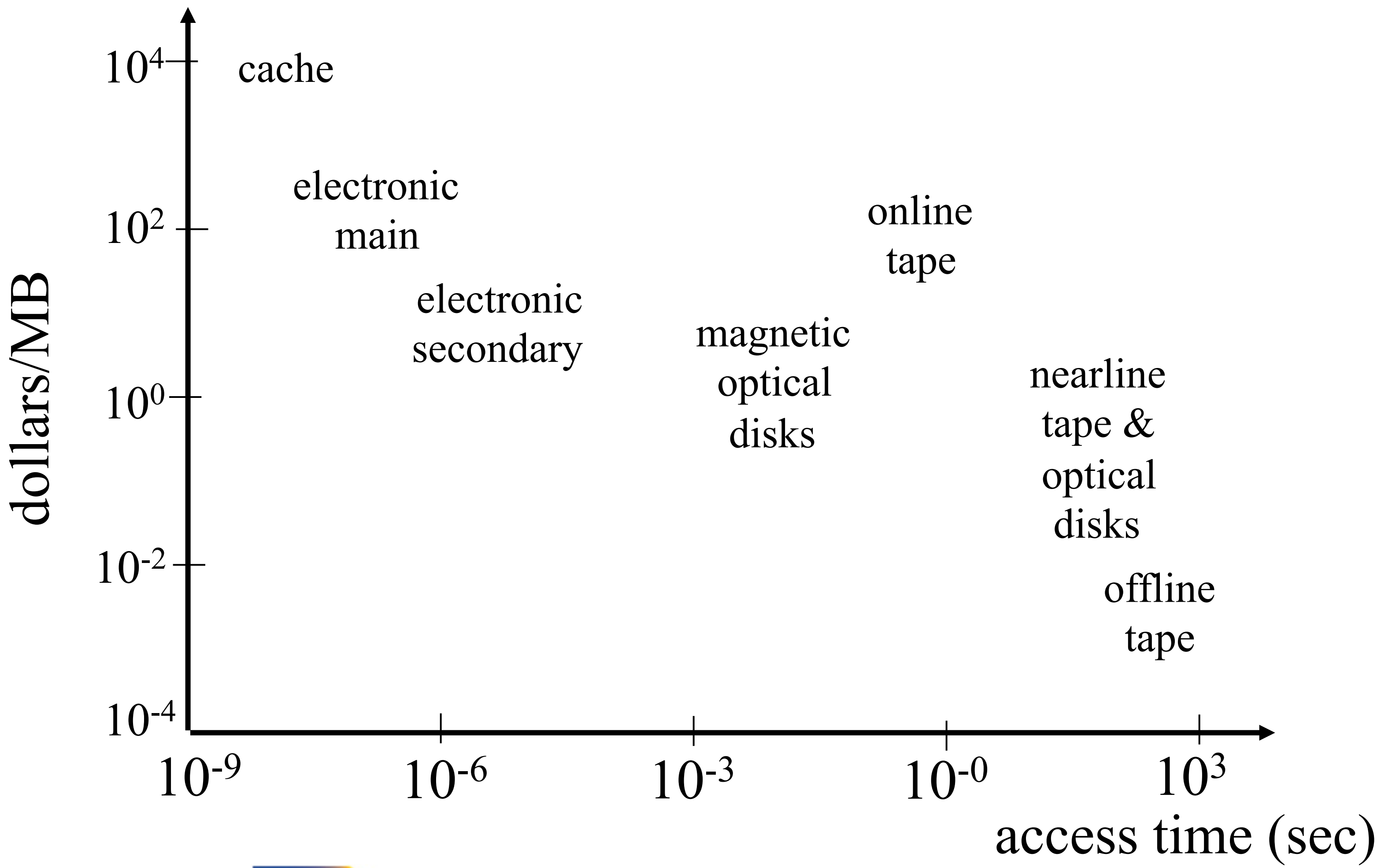
Storage Cost



from Gray & Reuter

Storage Cost

from Gray & Reuter



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

- THE 5 MINUTE RULE FOR TRADING MEMORY FOR DISC ACCESSES
  Jim Gray & Franco Putzolu
  May 1985

- The Five Minute Rule, Ten Years Later
  Goetz Graefe & Jim Gray
  December 1997

Five Minute Rule

- Say a page is accessed every X seconds
- CD = cost if we keep that page on disk
  - $D = cost of disk unit
  - I = numbers IOs that unit can perform per second
  - In X seconds, unit can do XI IOs
  - So CD = $D / XI

Five Minute Rule

- Say a page is accessed every X seconds
- CM = cost if we keep that page on RAM
  - $M = cost of 1 MB of RAM
  - P = numbers of pages in 1 MB RAM
  - So CM = $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

$$X = \frac{\$D \;\; P}{I \;\; \$M}$$

Using '97 Numbers

- P = 128 pages/MB (8KB pages)
- I = 64 accesses/sec/disk
- $D = 2000 dollars/disk (9GB + controller)
- $M = 15 dollars/MB of DRAM

- X = 266 seconds (about 5 minutes)
  (did not change much from 85 to 97)

Disk Failures

- Partial $\rightarrow$ Total
- Intermittent $\rightarrow$ Permanent

Coping with Disk Failures

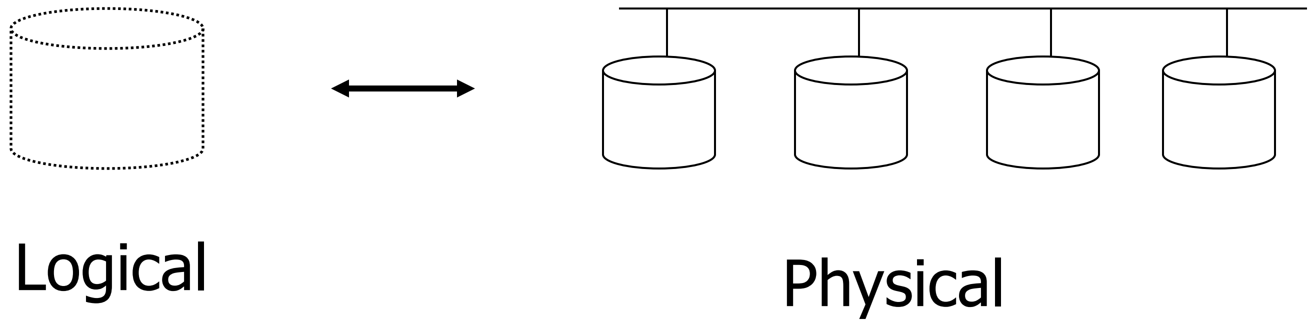
- Detection
  - e.g. Checksum

- Correction

  $\Rightarrow$ Redundancy

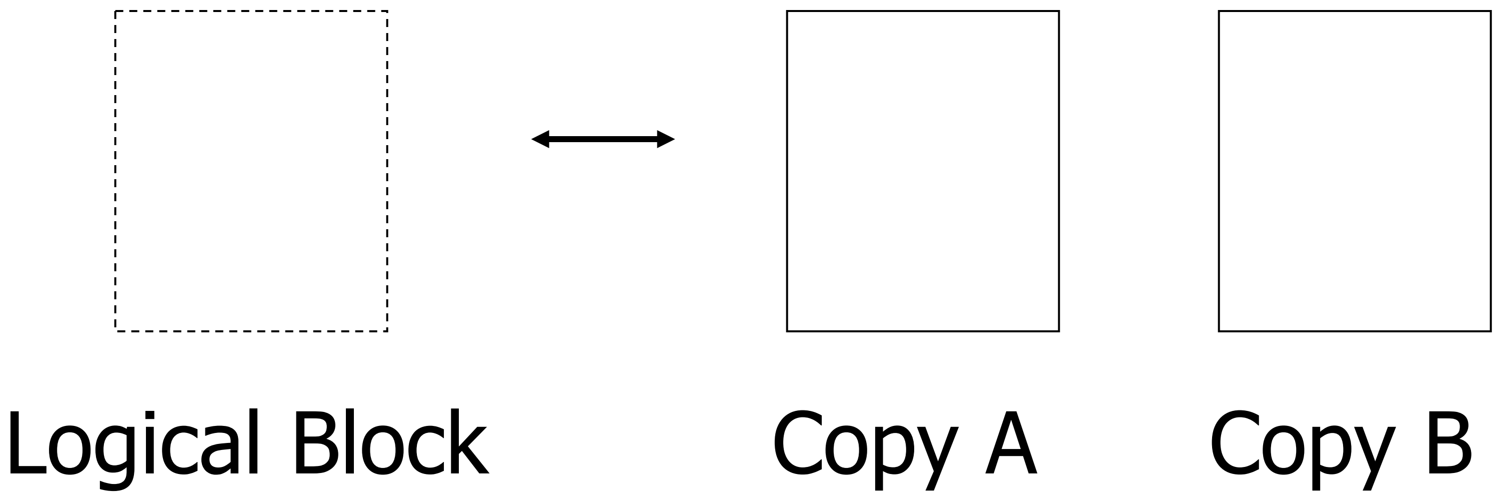
At what level do we cope?

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



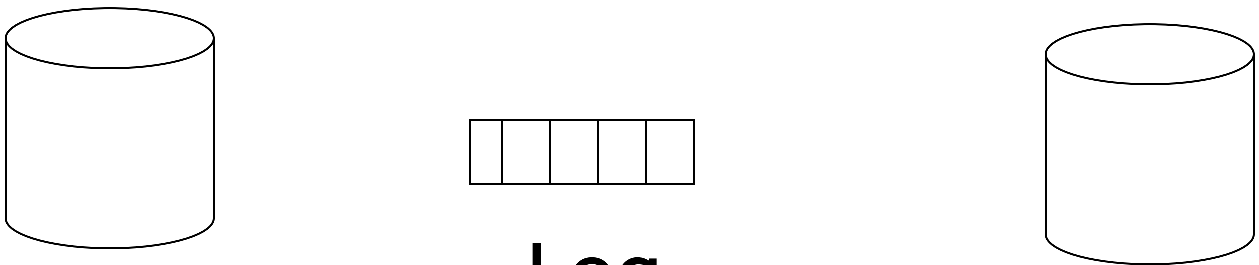
→ Operating System

e.g., Stable Storage



→ Database System

- e.g.,



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

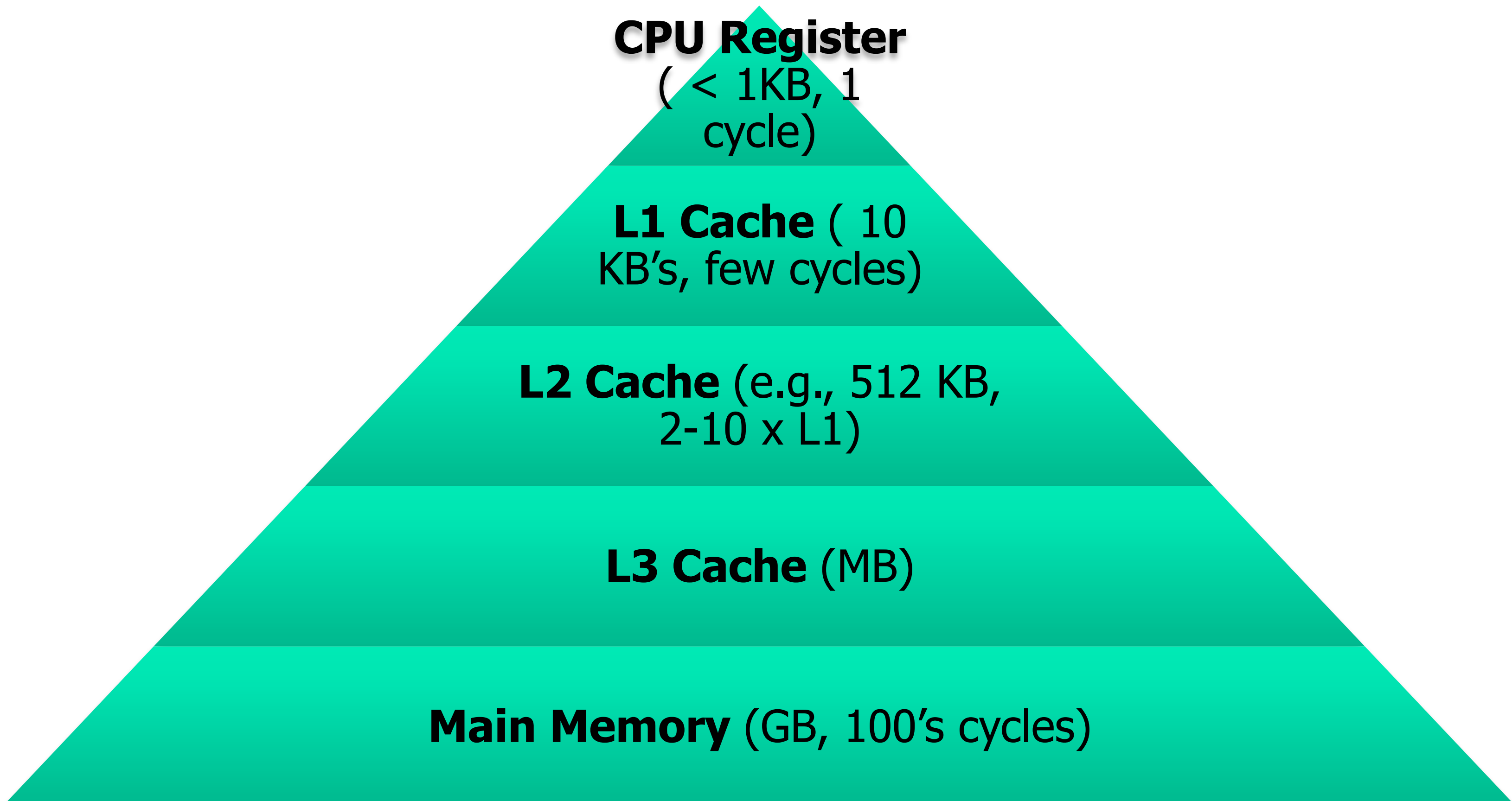
- Hardware: Disks
- Access Times
- Example: Megatron 747
- Optimizations
- Other Topics
  - Storage Costs
  - Using Secondary Storage
  - Disk Failures

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



Memory Hierarchy

- **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