

CSE 562

Database Systems

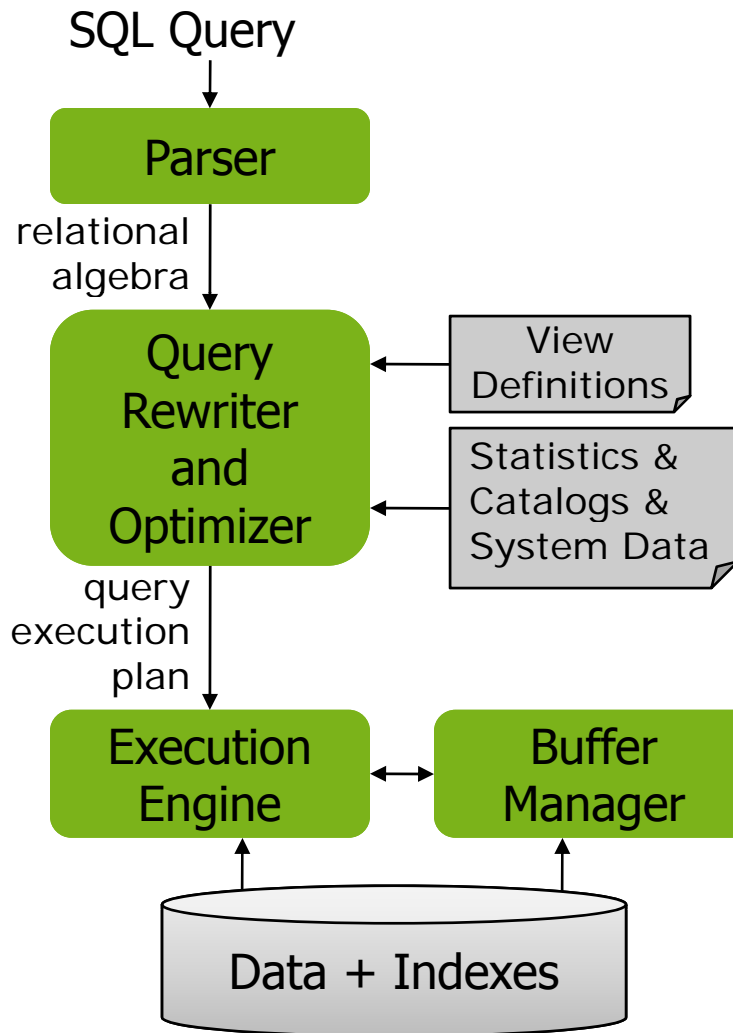
Hardware

Some slides are based or modified from originals by
Database Systems: The Complete Book,
Pearson Prentice Hall 2nd Edition
©2008 Garcia-Molina, Ullman, and Widom

cse@buffalo

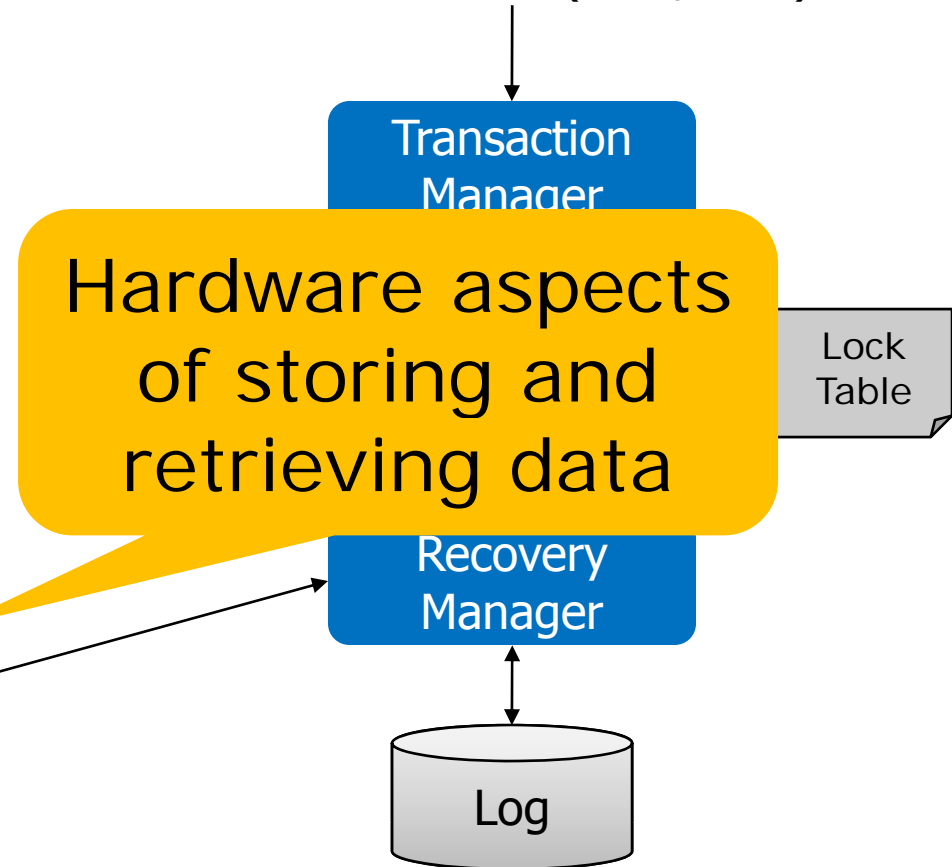
Database System Architecture

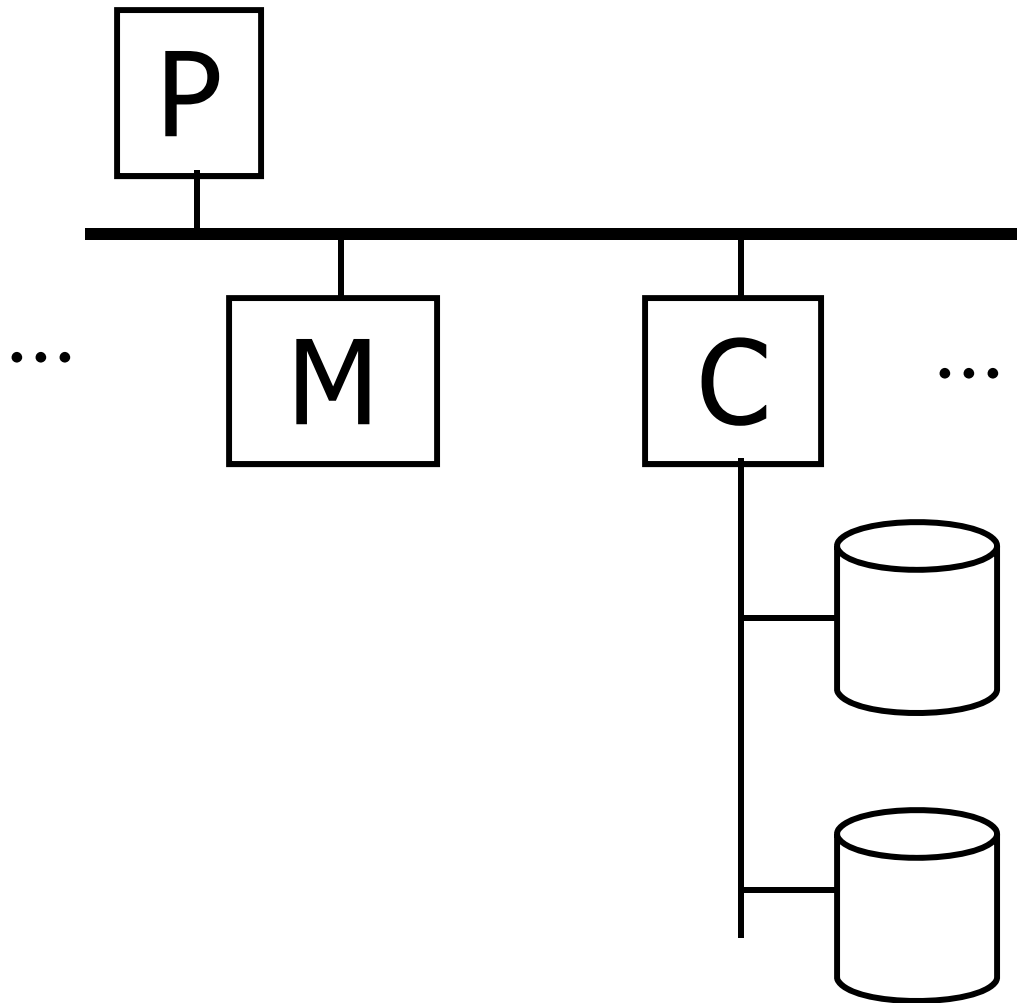
Query Processing



Transaction Management

Calls from Transactions (read,write)



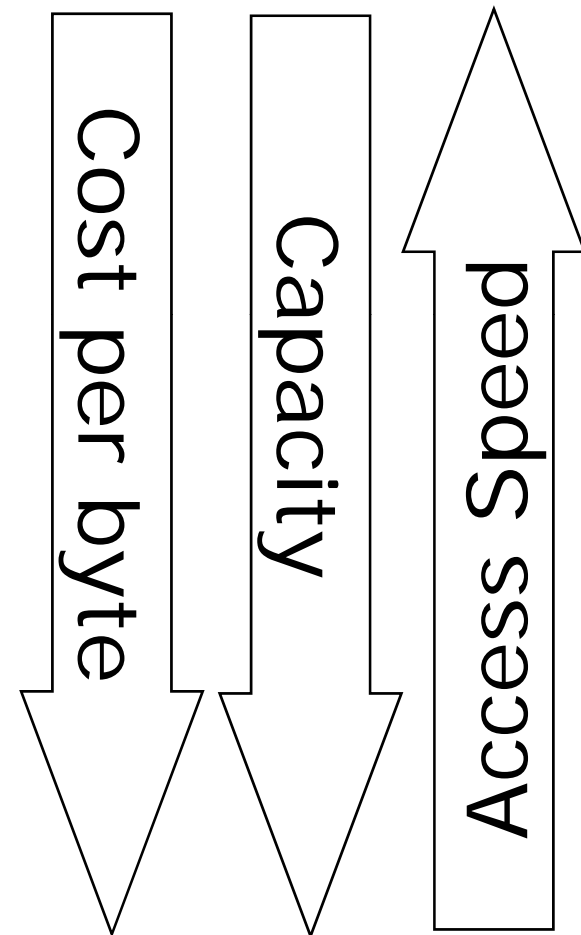


Typical
Computer

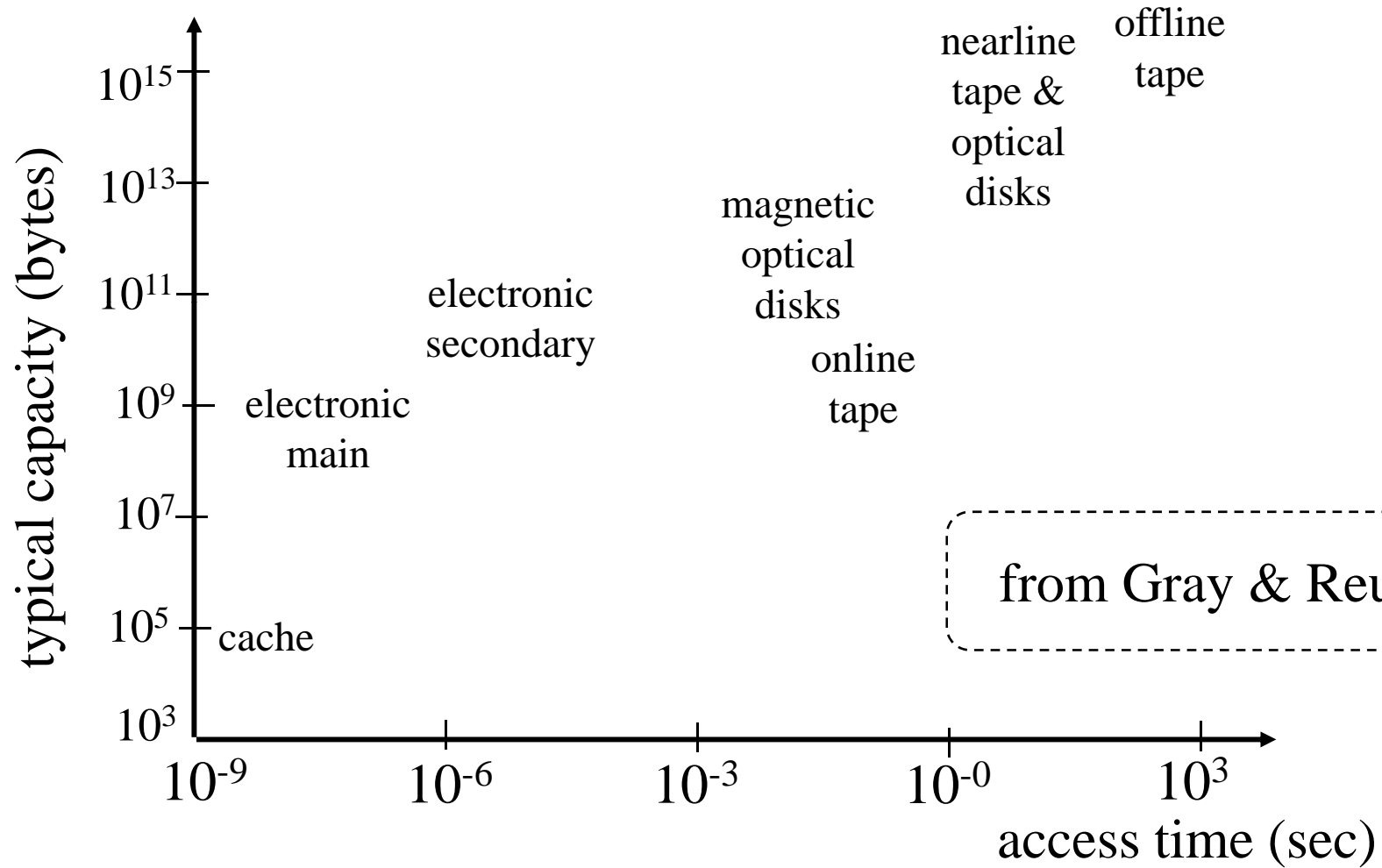
Secondary
Storage

Memory Hierarchy

- Cache memory
 - On-chip and L2
 - Caching outside control of DB system
- RAM
 - Addressable space includes virtual memory but DB systems avoid it
 - Main memory DBs rely more on OS
- Disk
 - Access speed & Transfer rate
 - Winchester, arrays,...
- Tertiary storage
 - Tapes, jukeboxes, DVDs

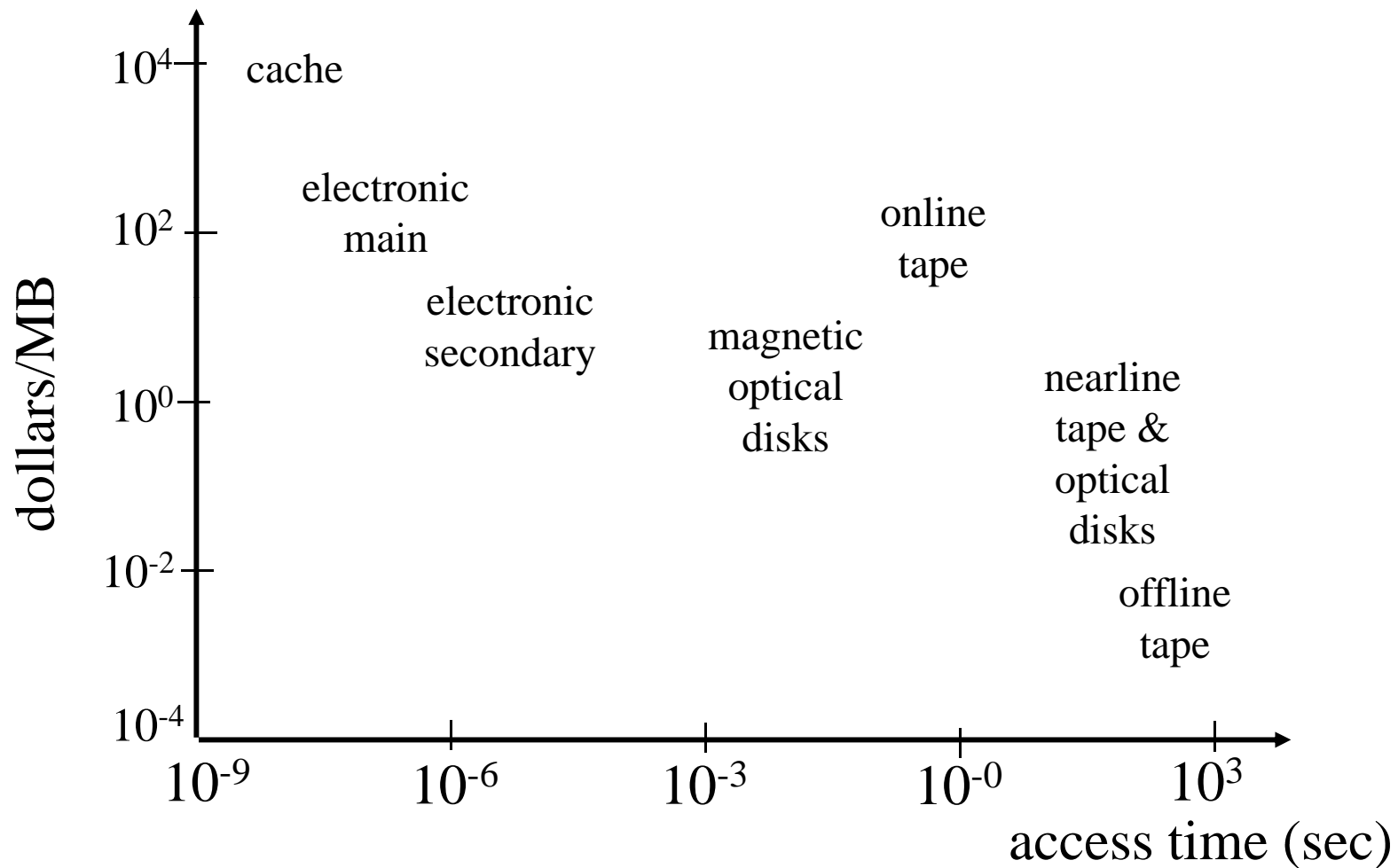


Storage Cost



Storage Cost

from Gray & Reuter



Volatile Vs Non-Volatile Storage

- Persistence important for transaction atomicity and durability
- Even if database fits in main memory changes have to be written in non-volatile storage
- Hard disk
- RAM disks w/ battery
- Flash memory

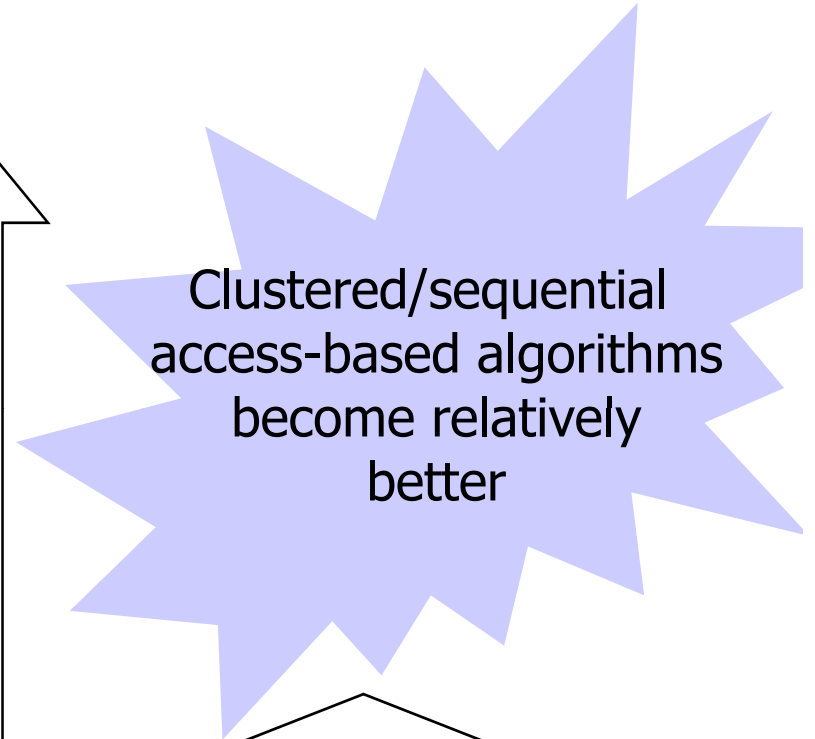
Cost of Disk Access

- How many blocks were accessed ?
- Clustered/consecutive ?

Moore's Law: Different Rates of Improvement

- Processor speed
- Main memory bit/\$
- Disk bit/\$
- RAM access speed
- Disk access speed
- Disk transfer rate

Disk Transfer Rate



Disk
Access
Time

Moore's Law: Different Rates of Improvement

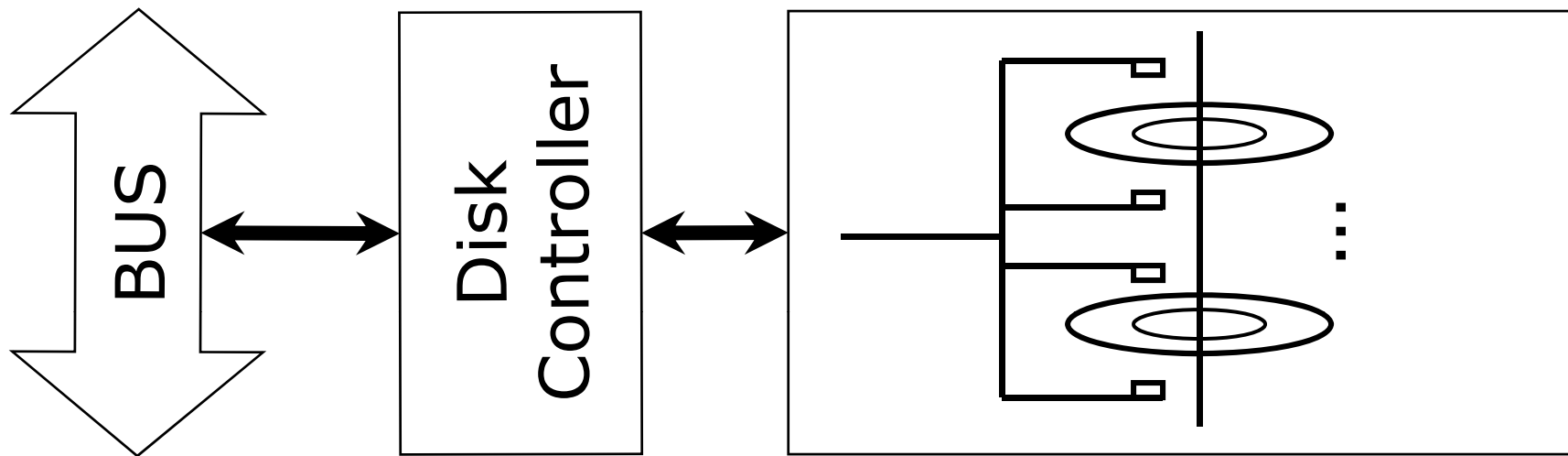
Cache Capacity

RAM Capacity

Cost of "miss"
increases

Disk
Access
Time

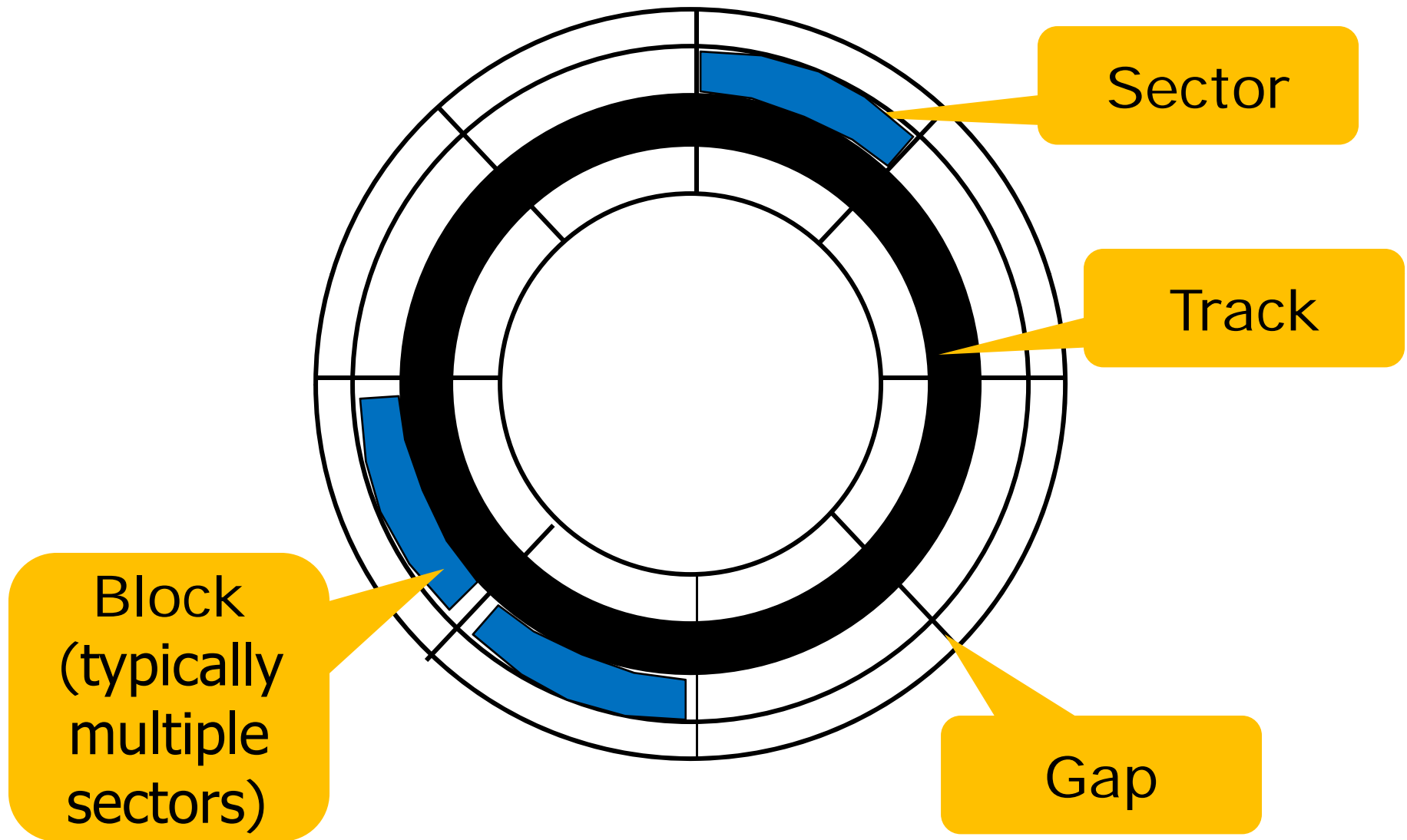
Focus on: "Typical Disk"



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

Top View

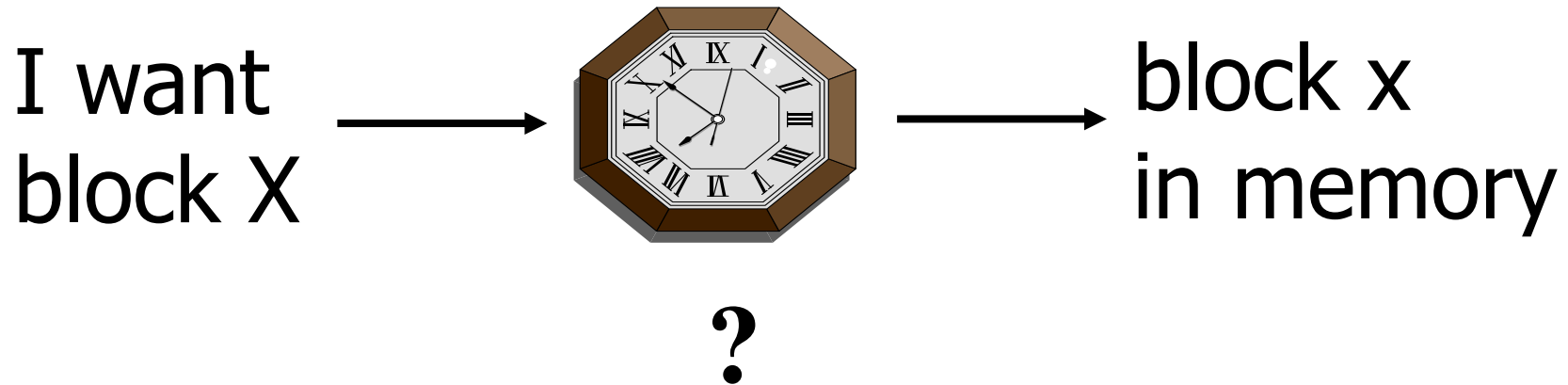
Often different numbers of sectors per track



“Typical” Numbers

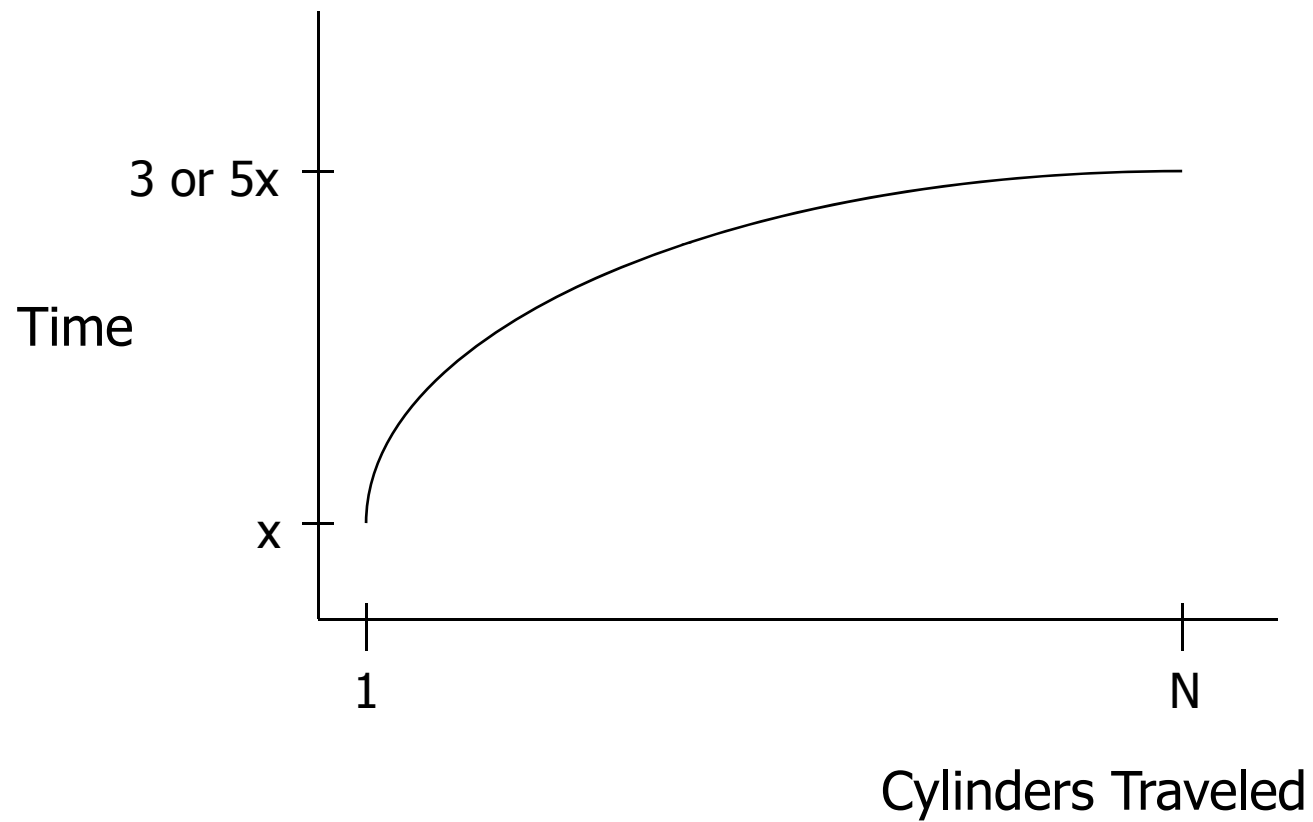
Diameter:	1 inch → 15 inches
Cylinders:	100 → 2000
Surfaces:	1 (CDs) →
(Tracks/cyl)	2 (floppies) → 30
Sector Size:	512B → 50K
Capacity:	360 KB (old floppy) → 400 GB (I use)

Key Performance Metric: Time to Fetch Block



Time = Seek Time (locate track) +
Rotational Delay (locate sector) +
Transfer Time (fetch block) +
Other (disk controller, ...)

Seek Time

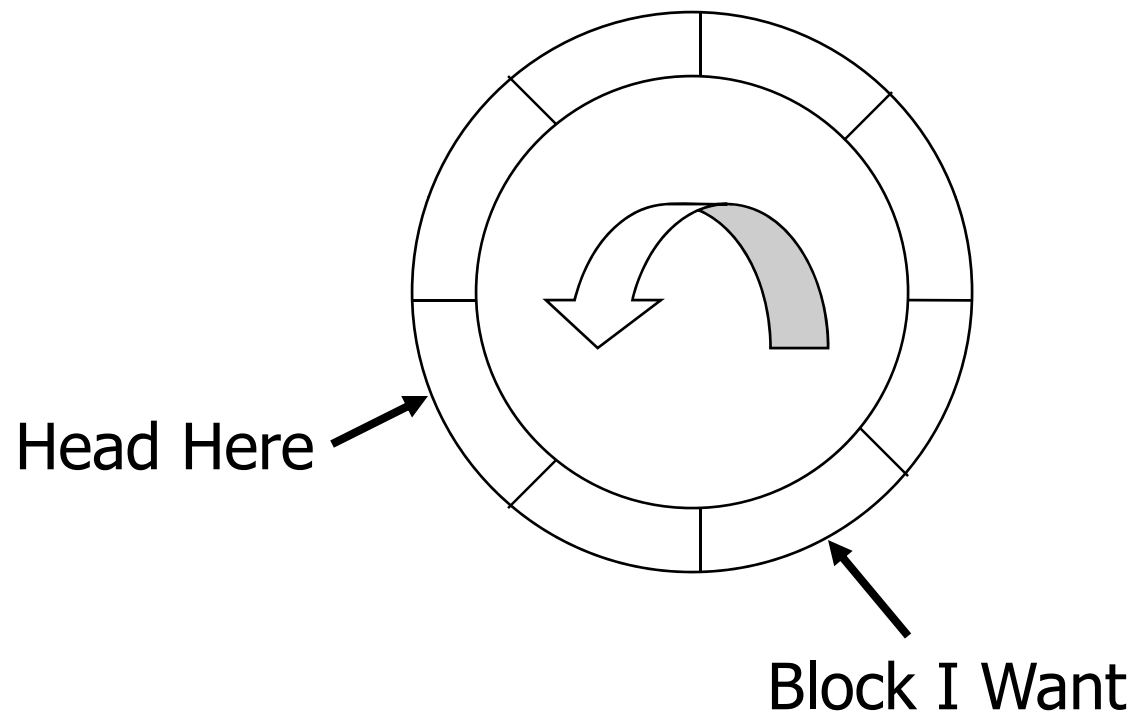


Average Random Seek Time

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

“Typical” S: 10 ms \rightarrow 40 ms

Rotational Delay



Average Rotational Delay

$R = 1/2$ revolution

“typical” $R = 8.33$ ms (7200 RPM)

Transfer Rate: t

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

Other Delays

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

“Typical” Value: 0

Practice Problem

- Single surface
- Rotation speed 7200rpm
- 16,384 tracks
- 128 sectors/track
- 4096 bytes/sector
- 4 sectors/block (16,384 bytes/block)
- SEEKTIME ($i \rightarrow j$) = $[1000 + (j-i)] \mu\text{s}$
- Neglect gaps
- Calculate minimum, maximum, average time to fetch one block

Practice Problem: Minimum Time

- Head is at the start of the first sector of the block
- Just compute transfer time
- 4 sectors cover $4/128$ of a track
- 1 full rotation takes $60/7200=8.33\text{ms}$
- Transfer time is $8.33 * 4 /128 = 0.26\text{ms}$

Practice Problem: Maximum Time

- Assume read must start at the first sector
- Head is at innermost, required track is the outermost
- Seek time = ...
- Head just missed the beginning
- Rotational delay = ...
- Transfer time = ...

Practice Problem: Average Time

- Solve...

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

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

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



- 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.

Practice Problem cont'd: Sustained Bandwidth over Track

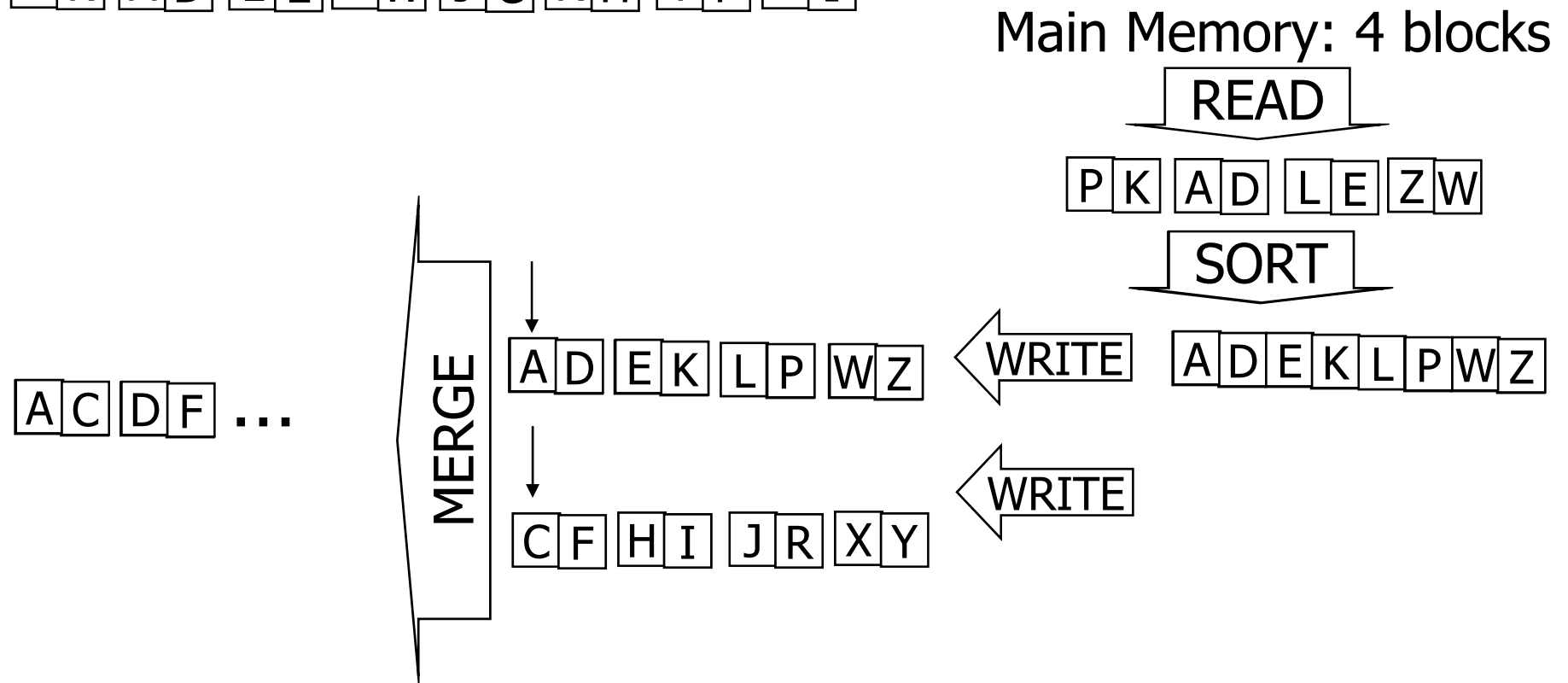
- Assume required blocks are consecutive on single track
- What is the sustained bandwidth of fetching consecutive blocks?
- 128 sectors/track * 4KB/sector in
8.33ms/track full rotation =
 $512\text{KB}/8.33\text{ms} = 61.46\text{KB/ms}$

Suggested Optimization

- Cluster data in consecutive blocks
- Give an extra point to algorithms that
 - exploit data clustering by avoiding “random” accesses
 - Read/write consecutive blocks

Example: 2-Phase Merge Sort

P K A D L E Z W J C R H Y F X I



Improve by bringing max number of blocks in memory

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 Block:

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

Block Address:

- Physical Device
- Cylinder #
- Surface #
- Sector

Once upon a time DBs
had access to such – now
it is the OS's domain

Optimizations (in controller or O.S.)

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

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 ...

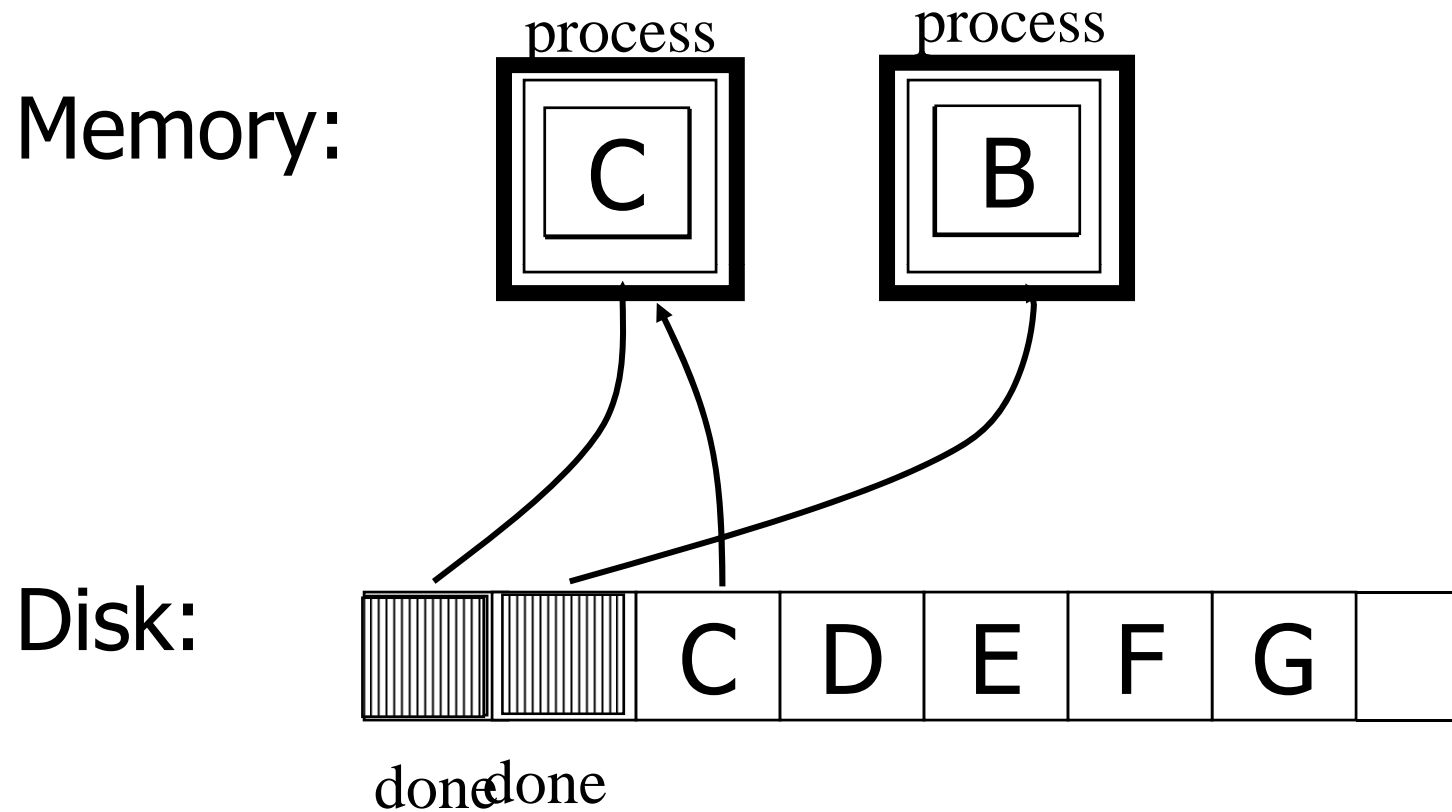
Say P = time to process/block

R = time to read in 1 block

n = # blocks

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

Double Buffering



Say $P \geq R$

P = Processing time/block

R = IO time/block

n = # blocks

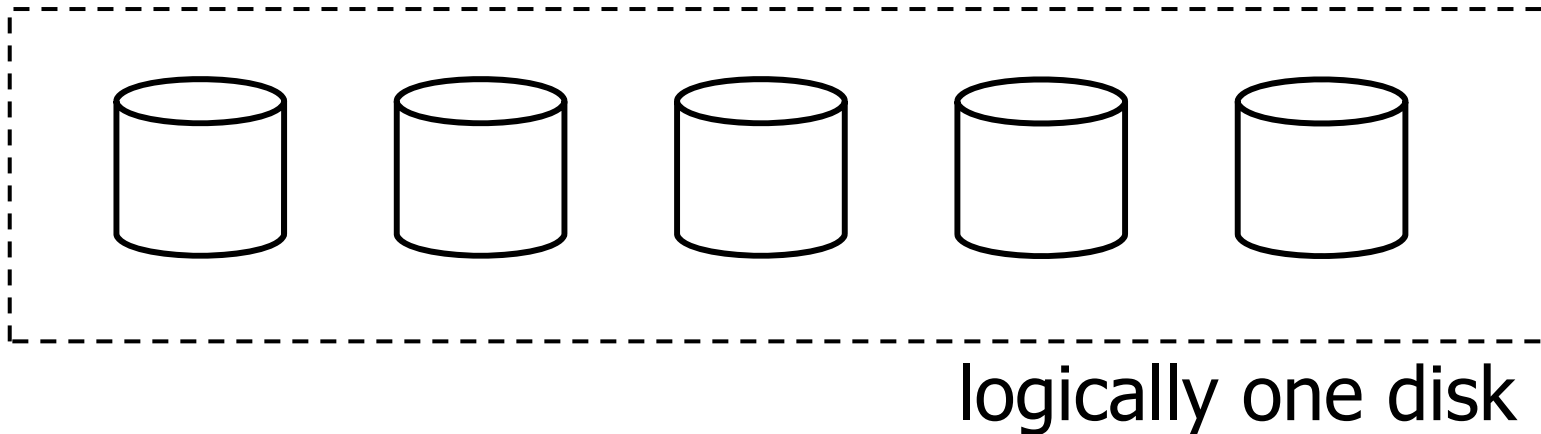
What is processing time?

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

Improvement much more dramatic if consecutive blocks...

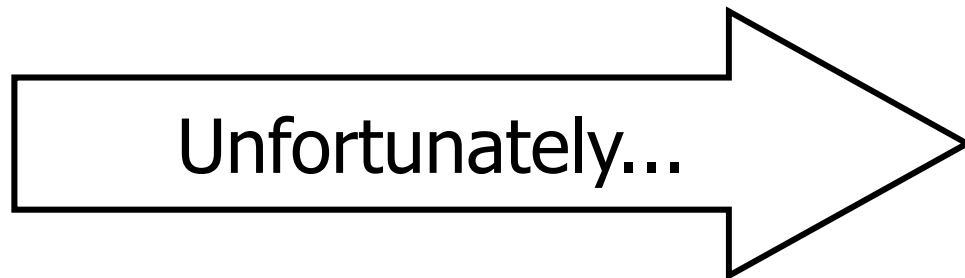
Disk Arrays

- RAIDs (various flavors)
- Block Striping
- Mirrored



Block Size Selection?

- Big Block → Amortize I/O Cost



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

Trend

- memory prices drop and memory capacities increase,
- transfer rates increase
- disk access times do not increase that much

⇒ blocks get bigger ...

Disk Failures

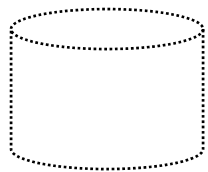
- Partial → Total
- Intermittent → Permanent

Coping with Disk Failures

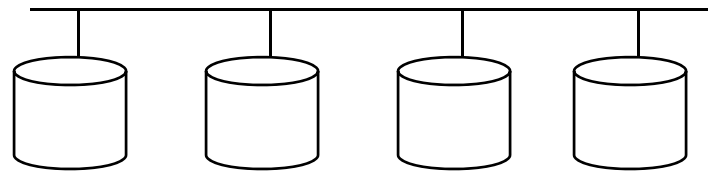
- Detection
 - e.g. Checksum
- Correction
 - ⇒ Redundancy

At what level do we cope?

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



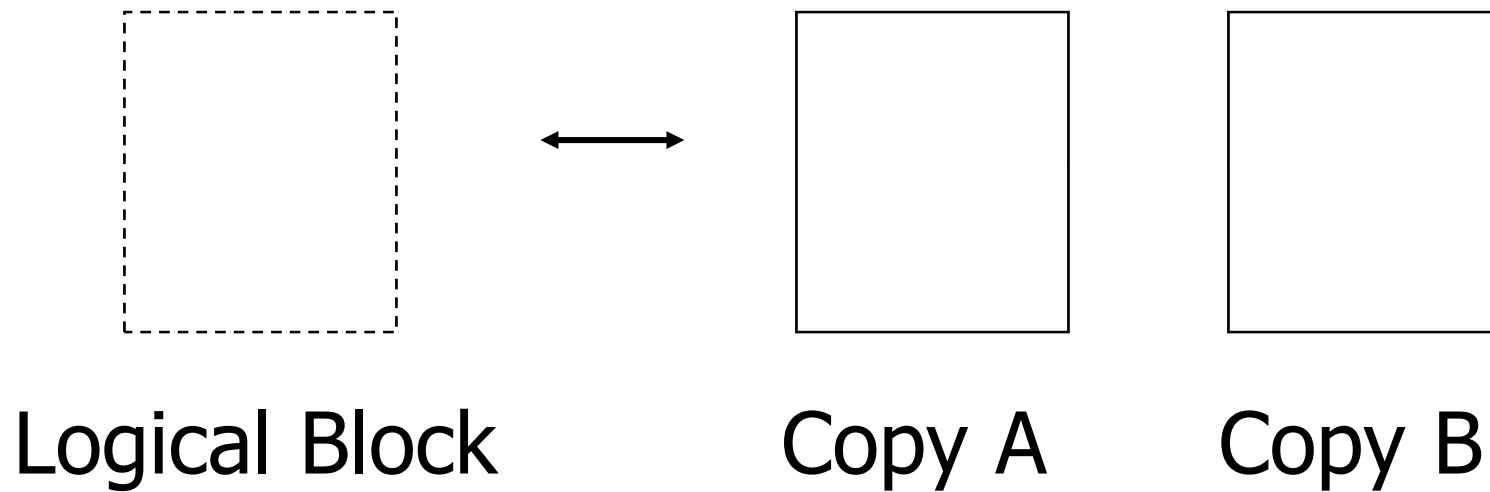
Logical



Physical

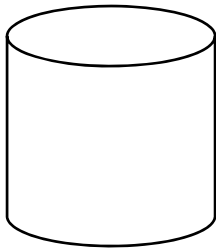
Operating System

e.g., Stable Storage



Database System

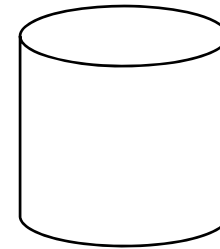
- e.g.,



Current DB



Log



Last week's DB

RAID

- **RAID: Redundant Arrays of Independent Disks**
Disk organization techniques that manage a large numbers of disks, providing a view of a single disk of
 - high capacity and high speed by using multiple disks in parallel, and
 - high reliability by storing data redundantly, so that data can be recovered even if a disk fails
- The chance that some disk out of a set of N disks will fail is much higher than the chance that a specific single disk will fail
 - E.g., a system with 100 disks, each with MTTF of 100,000 hours (approx. 11 years), will have a system MTTF of 1000 hours (approx. 41 days)
 - Techniques for using redundancy to avoid data loss are critical with large numbers of disks

Improvement of Reliability via Redundancy

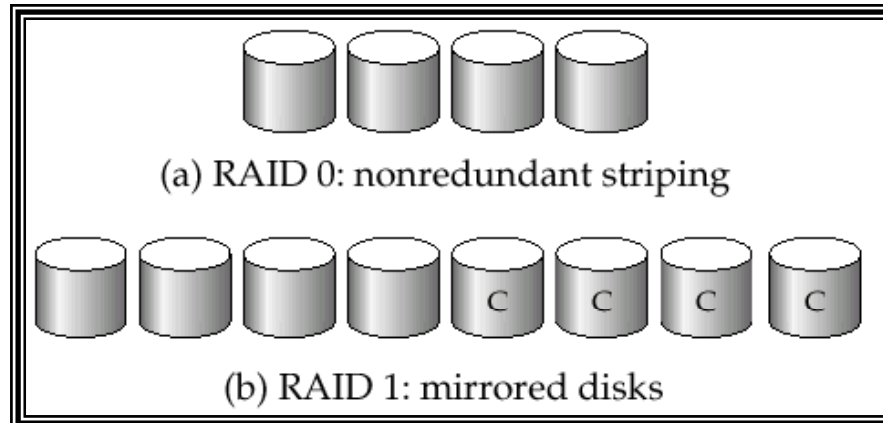
- **Redundancy** – store extra information that can be used to rebuild information lost in a disk failure
- E.g., **Mirroring** (or **shadowing**)
 - Duplicate every disk. Logical disk consists of two physical disks.
 - Every write is carried out on both disks
 - Reads can take place from either disk
 - If one disk in a pair fails, data still available in the other
 - Data loss would occur only if a disk fails, and its mirror disk also fails before the system is repaired
 - Probability of combined event is very small
- Mean time to data loss depends on mean time to failure, and mean time to repair
 - E.g. MTTF of 100,000 hours, mean time to repair of 10 hours gives mean time to data loss of 500×10^6 hours (or 57,000 years) for a mirrored pair of disks (ignoring dependent failure modes)

Improvement in Performance via Parallelism

- Two main goals of parallelism in a disk system:
 1. Load balance multiple small accesses to increase throughput
 2. Parallelize large accesses to reduce response time
- Improve transfer rate by striping data across multiple disks
- **Bit-level striping** – split the bits of each byte across multiple disks
 - In an array of eight disks, write bit i of each byte to disk i
 - Each access can read data at eight times the rate of a single disk
 - But seek/access time worse than for a single disk
 - Bit level striping is not used much any more
- **Block-level striping** – with n disks, block i of a file goes to disk $(i \bmod n) + 1$
 - Requests for different blocks can run in parallel if the blocks reside on different disks
 - A request for a long sequence of blocks can utilize all disks in parallel

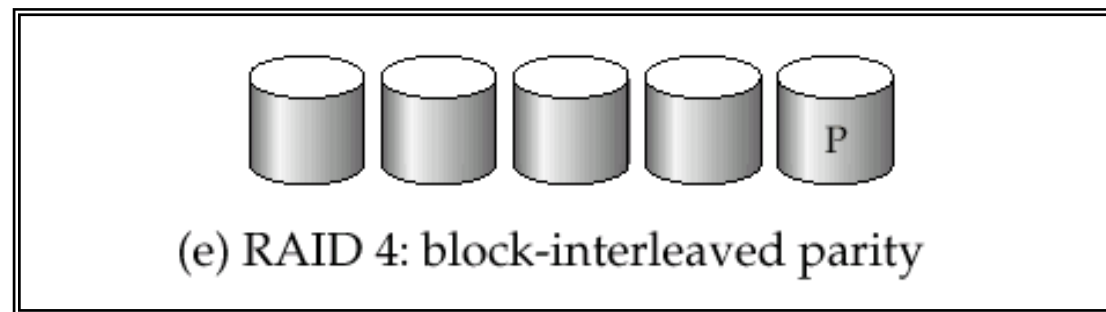
RAID Levels

- Schemes to provide redundancy at lower cost by using disk striping combined with parity bits
 - Different RAID organizations, or RAID levels, have differing cost, performance and reliability characteristics
- **RAID Level 0:** Block striping; non-redundant.
 - Used in high-performance applications where data loss is not critical
- **RAID Level 1:** Mirrored disks
 - RAID Level 1+0: Mirrored disks with block striping
 - Popular for applications such as storing log files in a database system



RAID Levels (cont'd)

- **RAID Level 4:** Block-Interleaved Parity; uses block-level striping, and keeps a parity block on a separate disk for corresponding blocks from N other disks
 - When writing data block, corresponding block of parity bits must also be computed and written to parity disk
 - To find value of a damaged block, compute XOR of bits from corresponding blocks (including parity block) from other disks

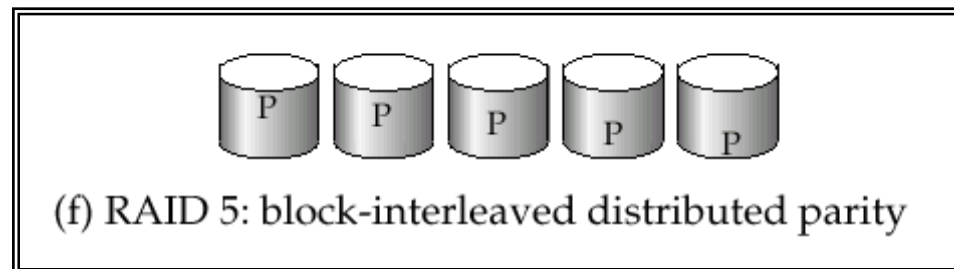


RAID Levels (cont'd)

- RAID Level 4 (Cont'd)
 - Provides high I/O rates for independent block reads
 - block read goes to a single disk, so blocks stored on different disks can be read in parallel
 - Provides high transfer rates for reads of multiple blocks than no-striping
 - Before writing a block, parity data must be computed
 - Can be done by using old parity block, old value of current block and new value of current block (2 block reads + 2 block writes)
 - Or by recomputing the parity value using the new values of blocks corresponding to the parity block
 - More efficient for writing large amounts of data sequentially
 - Parity block becomes a bottleneck for independent block writes since every block write also writes to parity disk

RAID Levels (cont'd)

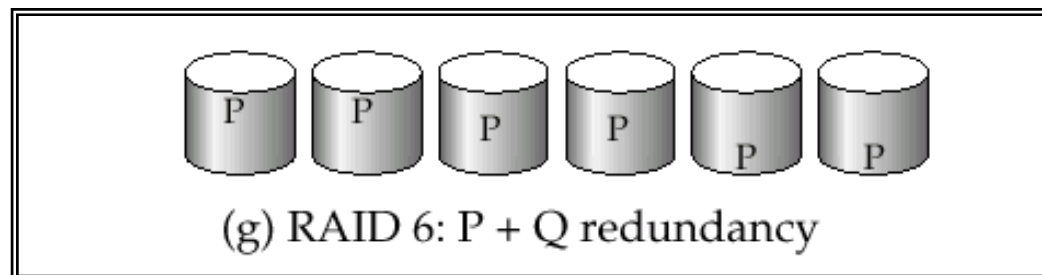
- **RAID Level 5: Block-Interleaved Distributed Parity;** partitions data and parity among all $N + 1$ disks, rather than storing data in N disks and parity in 1 disk
 - E.g., with 5 disks, parity block for n th set of blocks is stored on disk $(n \bmod 5) + 1$, with the data blocks stored on the other 4 disks



P0	0	1	2	3
4	P1	5	6	7
8	9	P2	10	11
12	13	14	P3	15
16	17	18	19	P4

RAID Levels (cont'd)

- **RAID Level 5 (cont.)**
 - Higher I/O rates than Level 4
 - Block writes occur in parallel if the blocks and their parity blocks are on different disks
 - Subsumes Level 4: provides same benefits, but avoids bottleneck of parity disk
- **RAID Level 6: P+Q Redundancy scheme; similar to Level 5, but stores extra redundant information to guard against multiple disk failures**
 - Better reliability than Level 5 at a higher cost; not used as widely



Choice of RAID Level

- Factors in choosing RAID level
 - Monetary cost
 - Performance: Number of I/O operations per second, and bandwidth during normal operation
 - Performance during failure
 - Performance during rebuild of failed disk
 - Including time taken to rebuild failed disk
- RAID 0 is used only when data safety is not important
 - E.g. data can be recovered quickly from other sources
- Level 4 never used since it is subsumed by 5
- Level 6 is rarely used since levels 1 and 5 offer adequate safety for almost all applications
- So competition is between 1 and 5 only

Choice of RAID Level (cont'd)

- Level 1 provides much better write performance than level 5
 - Level 5 requires at least 2 block reads and 2 block writes to write a single block, whereas Level 1 only requires 2 block writes
 - Level 1 preferred for high update environments such as log disks
- Level 1 had higher storage cost than level 5
 - disk drive capacities increasing rapidly (50%/year) whereas disk access times have decreased much less (x 3 in 10 years)
 - I/O requirements have increased greatly, e.g. for Web servers
 - When enough disks have been bought to satisfy required rate of I/O, they often have spare storage capacity
 - so there is often no extra monetary cost for Level 1!
- Level 5 is preferred for applications with low update rate, and large amounts of data
- Level 1 is preferred for all other applications

Summary

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

This Time

- Hardware
 - Chapter 13: 13.1-13.4

Next Time

- Hardware
 - Chapter 13: 13.5-13.8