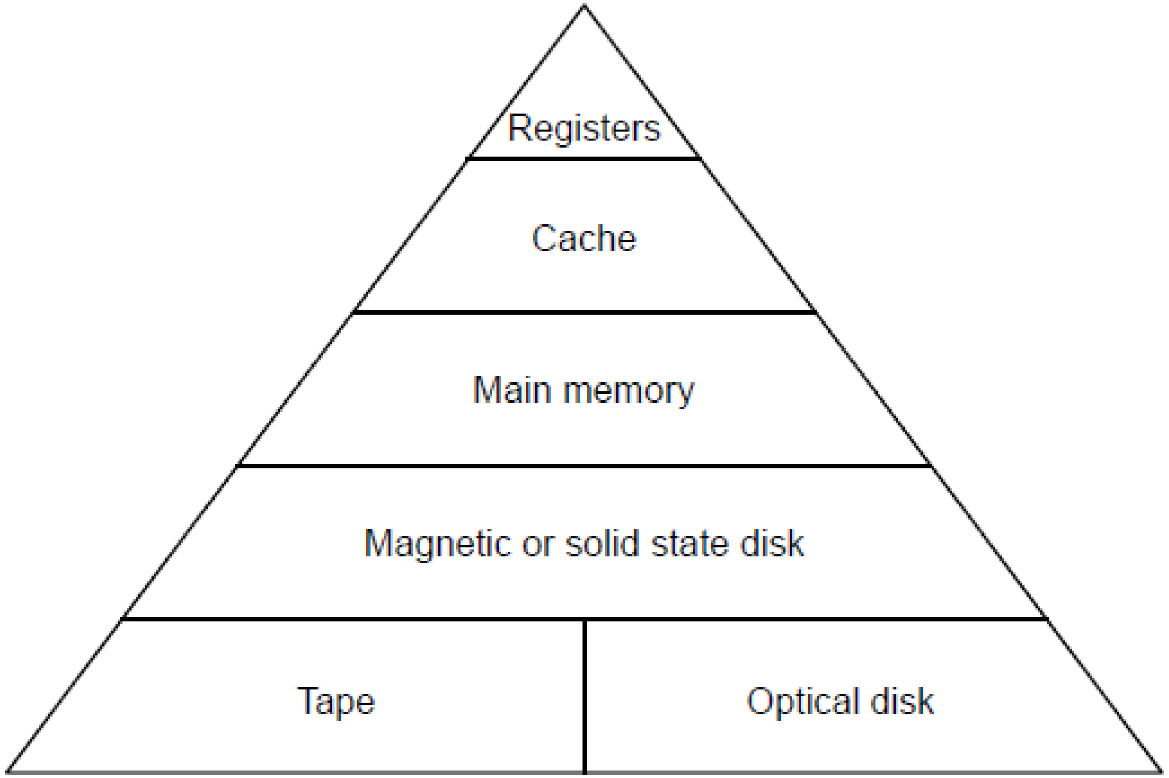
Input/Output

Chapter 5

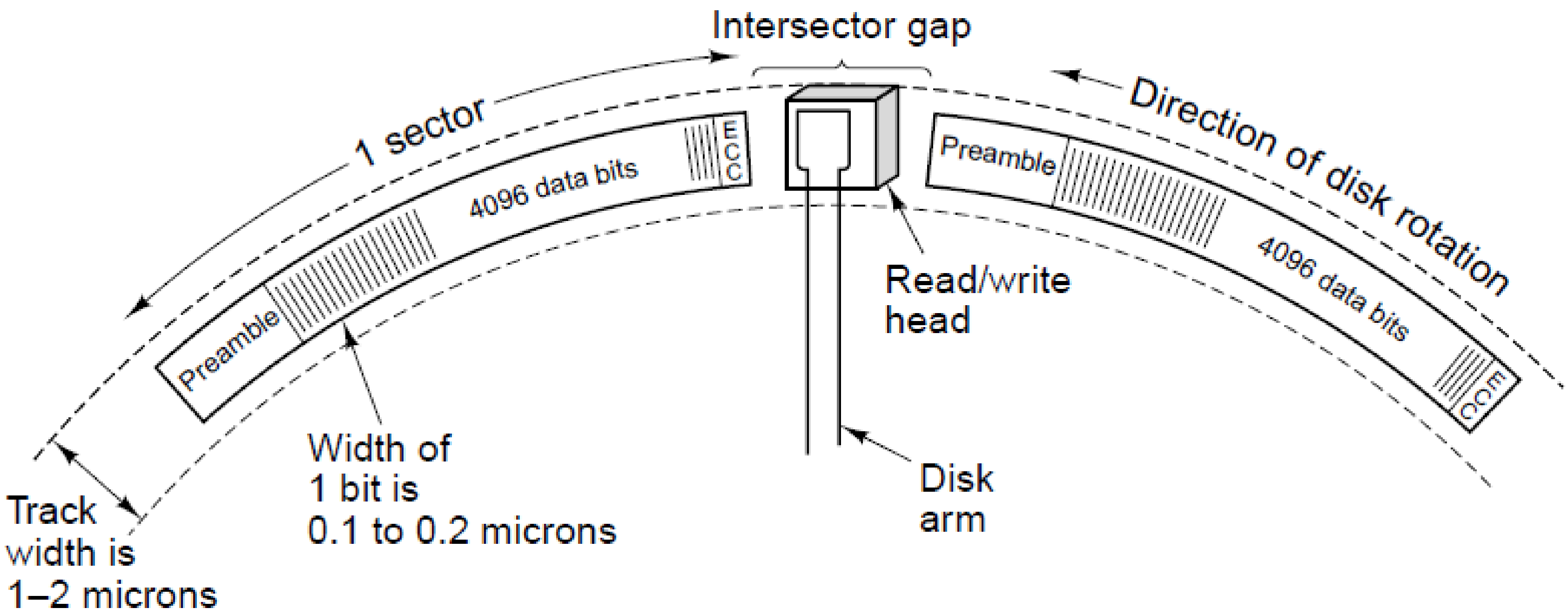
Secondary Memory: Memory Hierarchies

- Figure 2-18. A

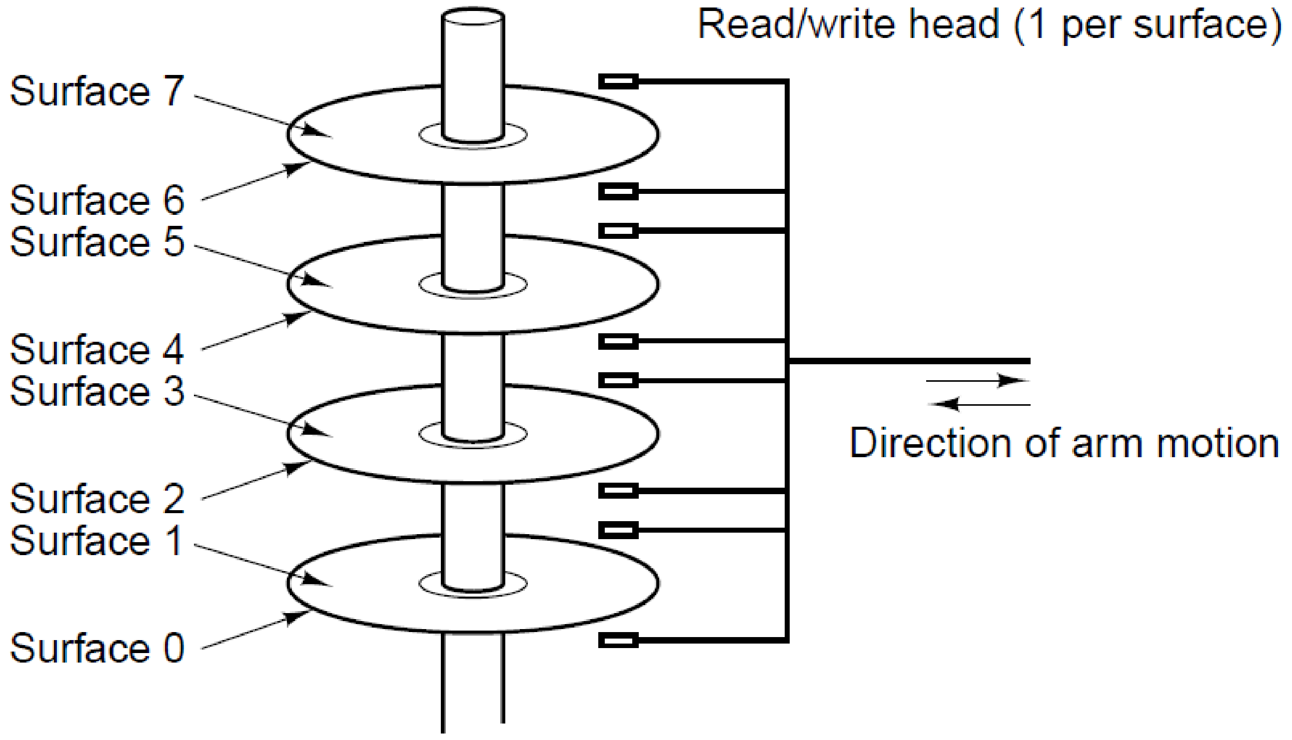


Magnetic Disks (1)

- Figure 2-19



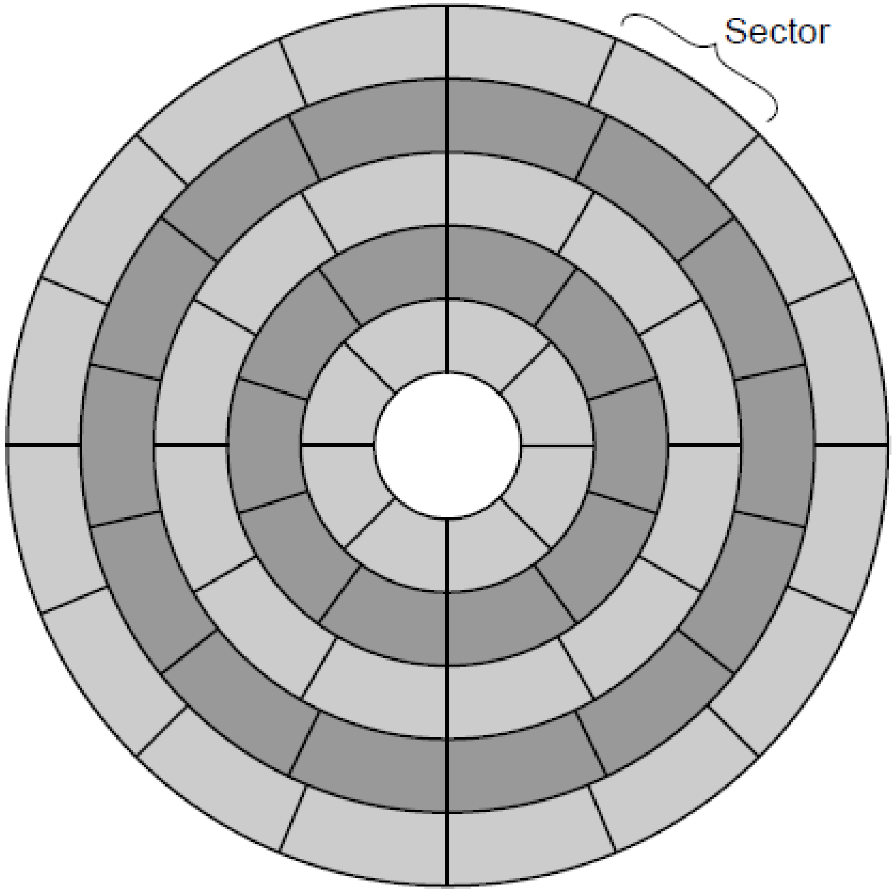
Magnetic Disks (2)



Magnetic disks are organized into cylinders.

- Figure 2-20. A disk with four platters.

Magnetic Disks (3)



- Figure 2-21. A disk with five zones. Each zone has many tracks.

I/O Devices

- I/O units often consist of electronic component (device controller/adapter) and mechanical component (device itself).
- I/O devices can roughly be divided into two categories: block devices and character devices.
  - **Block devices**:
  - stores information in fixed-size blocks, each with its own address. Common block sizes range from 512 to 65, 536 bytes.
  - Hard disks, Blue-ray discs, and USB sticks are common block devices.
  - **Character devices**: delivers or accepts a stream of characters, without regard to any block structure. It is not addressable and does not have any seek operation.
  - Printers, network interfaces, mice are common examples of character devices,

Common Disk Interfaces to Connect Storage Devices to Computer System Bus

- ST-506 → ATA → IDE (Also called Parallel ATA or PATA)→ SATA (Serial Advanced Technology Attachment)
  - Ancient standard
  - Commands (read/write) and addresses in cylinder/head/sector format placed in device registers
  - Recent versions support Logical Block Addresses (LBA)
- SCSI (Small Computer Systems Interface)
  - Packet based, like TCP/IP
  - Device translates LBA to internal format (e.g. c/h/s)
  - Transport independent
    - USB drives, CD/DVD/Blu ray, Firewire
    - iSCSI is SCSI over TCP/IP and Ethernet

Small Computer System Interface (SCSI)

- A set of standards for physically connecting and transferring data between computers and peripheral devices.
- The SCSI standards define commands, protocols, electrical and optical interfaces.
- SCSI is most commonly used for hard disk drives and tape drives, but it can connect a wide range of other devices, including scanners and CD drives.

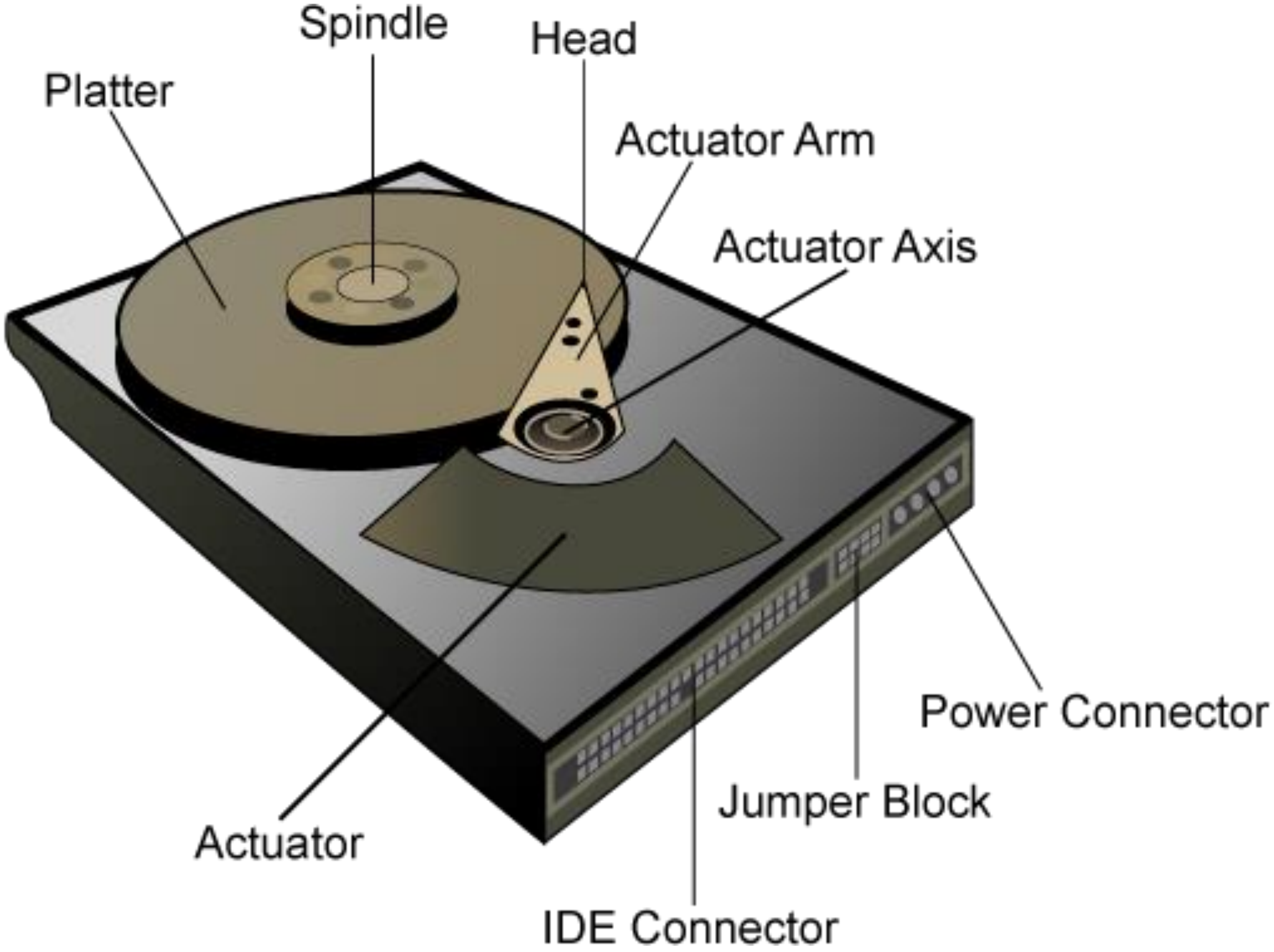
SCSI Disks

Name	Data bits	Bus MHz	MB/sec
SCSI-1	8	5	5
Fast SCSI	8	10	10
Wide Fast SCSI	16	10	20
Ultra SCSI	8	20	20
Wide Ultra SCSI	16	20	40
Ultra2 SCSI	8	40	40
Wide Ultra2 SCSI	16	40	80
Wide Ultra3 SCSI	16	80	160
Wide Ultra4 SCSI	16	160	320
Wide Ultra5 SCSI	16	320	640

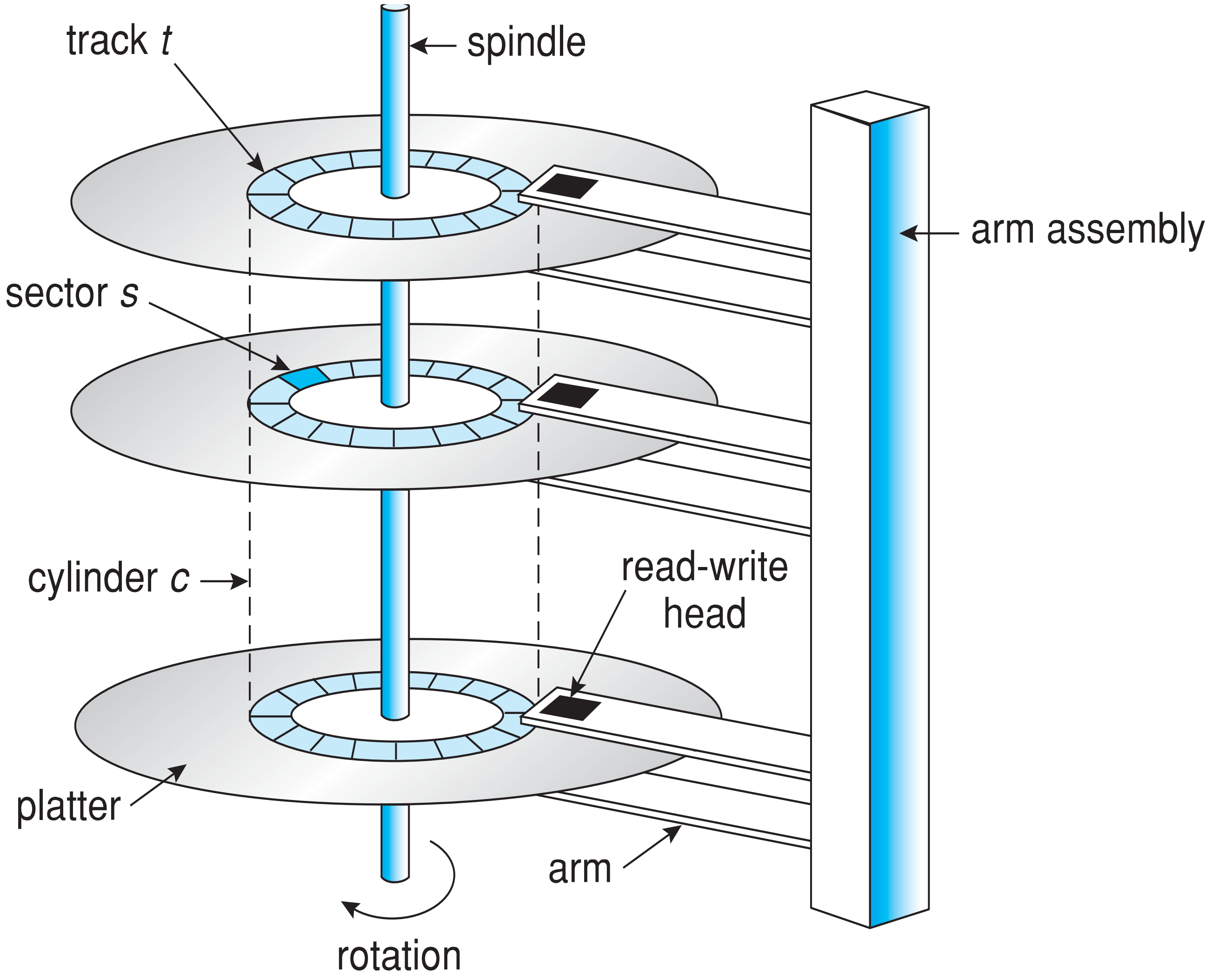
Storage Devices

- Hard Drives
- RAID
- SSD

Hard Drive Hardware



A Multi-Platter Disk



Addressing and Geometry

- Externally, hard drives expose a large number of sectors (blocks)
  - Typically 512 or 4096 bytes
  - Individual sector writes are atomic
  - Multiple sectors writes may be interrupted (torn write)
- Drive geometry
  - Sectors arranged into tracks
  - A cylinder is a particular track on multiple platters
  - Tracks arranged in concentric circles on platters
  - A disk may have multiple, double-sided platters
- Drive motor spins the platters at a constant rate
  - Measured in revolutions per minute (RPM)

Types of Delay With Disks

- **Three types of delay**

1. Rotational Delay
   - Time to rotate the desired sector to the read head
   - Related to RPM
2. Seek delay
   - Time to move the read head to a different track
3. Transfer time
   - Time to read or write bytes

How To Calculate Transfer Time

	Cheetah 15K.5	Barracuda
Capacity	300 GB	1 TB
RPM	15000	7200
Avg. Seek	4 ms	9 ms
Max Transfer	125 MB/s	105 MB/s

Transfer time

$$T_{I/O} = T_{seek} + T_{rotation} + T_{transfer}$$

Assume we are transferring 4096 bytes

Cheetah

$$T_{I/O} = 4\ ms + 1 / (15000\ RPM / 60\ s/M / 1000\ ms/s) / 2 + (4096\ B / 125\ MB/s * 1000\ ms/s / 2^{20}\ MB/B)$$

$$T_{I/O} = 4\ ms + 2ms + 0.03125\ ms \approx 6\ ms$$

Barracuda

$$T_{I/O} = 9\ ms + 1 / (7200\ RPM / 60\ s/M / 1000\ ms/s) / 2 + (4096\ B / 105\ MB/s * 1000\ ms/s / 2^{20}\ MB/B)$$

$$T_{I/O} = 9\ ms + 4.17\ ms + 0.0372\ ms \approx 13.2\ ms$$

Sequential vs. Random Access

- **Rate of I/O**
- $R_{I/O}$ = transfer_size / $T_{I/O}$

Access Type	Transfer Size		Cheetah 15K.5	Barracuda
Random	4096 B	$T_{I/O}$	6 ms	13.2 ms

Random I/O results in very poor disk performance!

Caching

- Many disks incorporate caches (track buffer)
  - Small amount of RAM (8, 16, or 32 MB)
- Read caching
  - Reduces read delays due to seeking and rotation
- Write caching
  - Write back cache: drive reports that writes are complete after they have been cached
    - Possibly dangerous feature. Why?
  - Write through cache: drive reports that writes are complete after they have been written to disk
- Today, some disks include flash memory for persistent caching (hybrid drives)

Disk Scheduling

- Caching helps improve disk performance
- But it can’t make up for poor random access times
- Key idea: if there are a queue of requests to the disk, they can be reordered to improve performance
  - First come, first serve (FCFC)
  - Shortest seek time first (SSTF)
  - SCAN, otherwise known as the elevator algorithm
  - C-SCAN, C-LOOK, etc.

Beyond Single Disks

- Hard drives are great devices
  - Relatively fast, persistent storage
- Shortcomings:
  - How to cope with disk failure?
    - Mechanical parts break over time
    - Sectors may become silently corrupted
  - Capacity is limited
    - Managing files across multiple physical devices is cumbersome
    - Can we make 10x 1 TB drives look like a 10 TB drive?

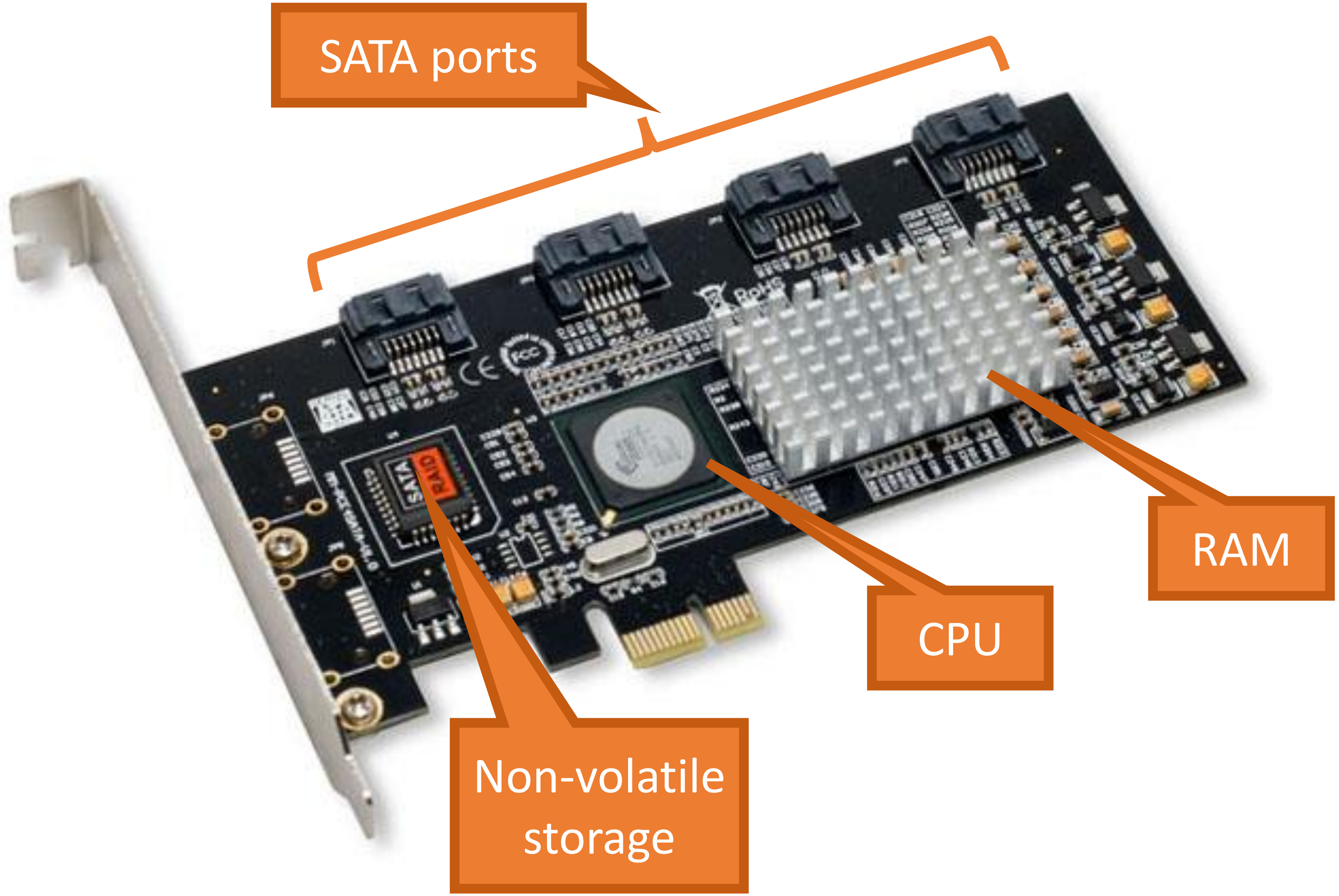
Redundant Array of Inexpensive Disks (RAID)

- Original Motivation:
  - Replacing large and expensive mainframe hard drives (IBM 3310) by several cheaper Winchester disk drives
- Today’s Motivation:
  - “Cheap” SCSI hard drives are now big enough for most applications
  - We use RAID today for
    - Increasing disk throughput by allowing parallel access
    - Eliminating the need to make disk backups
      - Disks are too big to be backed up in an efficient fashion

Redundant Array of Inexpensive Disks (RAID)

- RAID: use multiple disks to create the illusion of a large, faster, more reliable disk
- Externally, RAID looks like a single disk
  - i.e. RAID is transparent
  - Data blocks are read/written as usual
  - No need for software to explicitly manage multiple disks or perform error checking/recovery
- Internally, RAID is a complex computer system
  - Disks managed by a dedicated CPU + software
  - RAM and non-volatile memory
  - Many different configuration options (RAID levels)

Example RAID Controller



Mass Storage

- Many systems today need to store several terabytes of data
- Don’t want to use single, large disk
  - Too expensive
  - Failures could be catastrophic

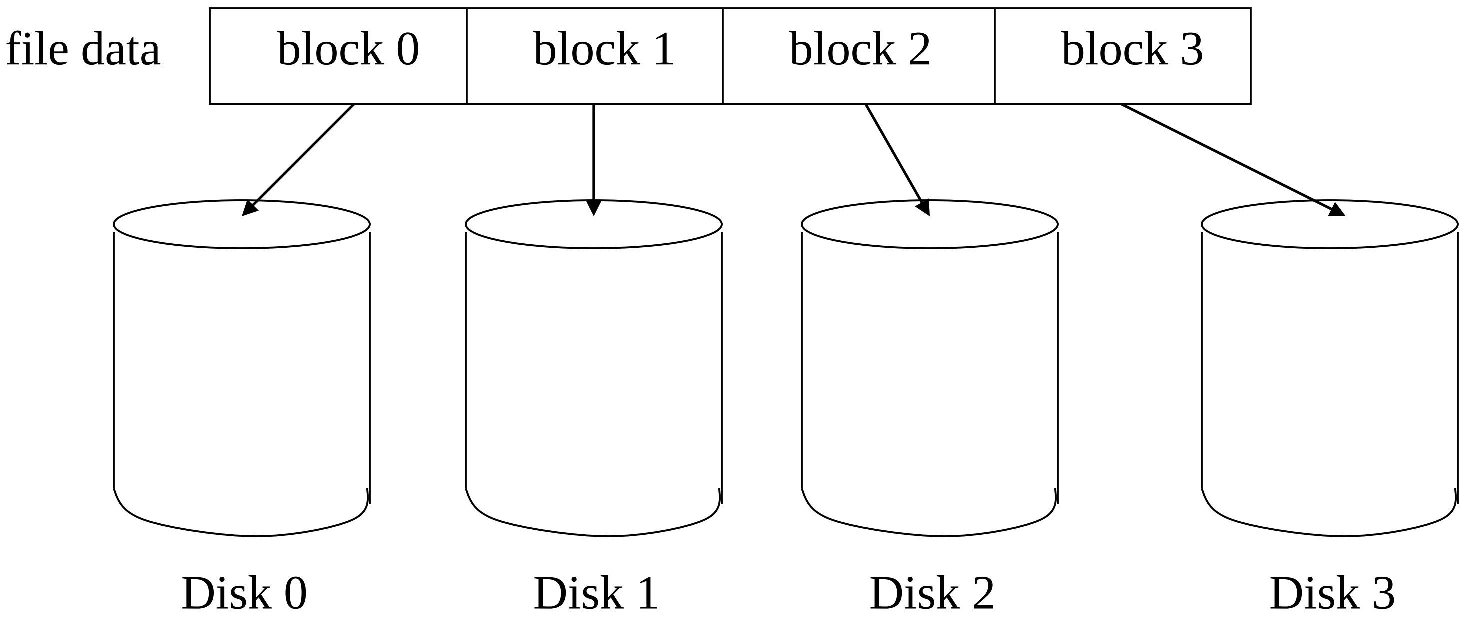
- Would prefer to use many smaller disks

RAID

- Redundant Array of Inexpensive Disks
- Basic idea is to connect multiple disks together to provide
  - large storage capacity
  - faster access to reading data
  - redundant data
- Many different levels of RAID systems
  - differing levels of redundancy, error checking, capacity, and cost

Striping

- Take file data and map it to different disks
- Allows for reading data in parallel



Parity

- Way to do error checking and correction
- Add up all the bits that are 1
  - if even number, set parity bit to 0
  - if odd number, set parity bit to 1
- To actually implement this, do an exclusive OR of all the bits being considered
- Consider the following 2 bytes

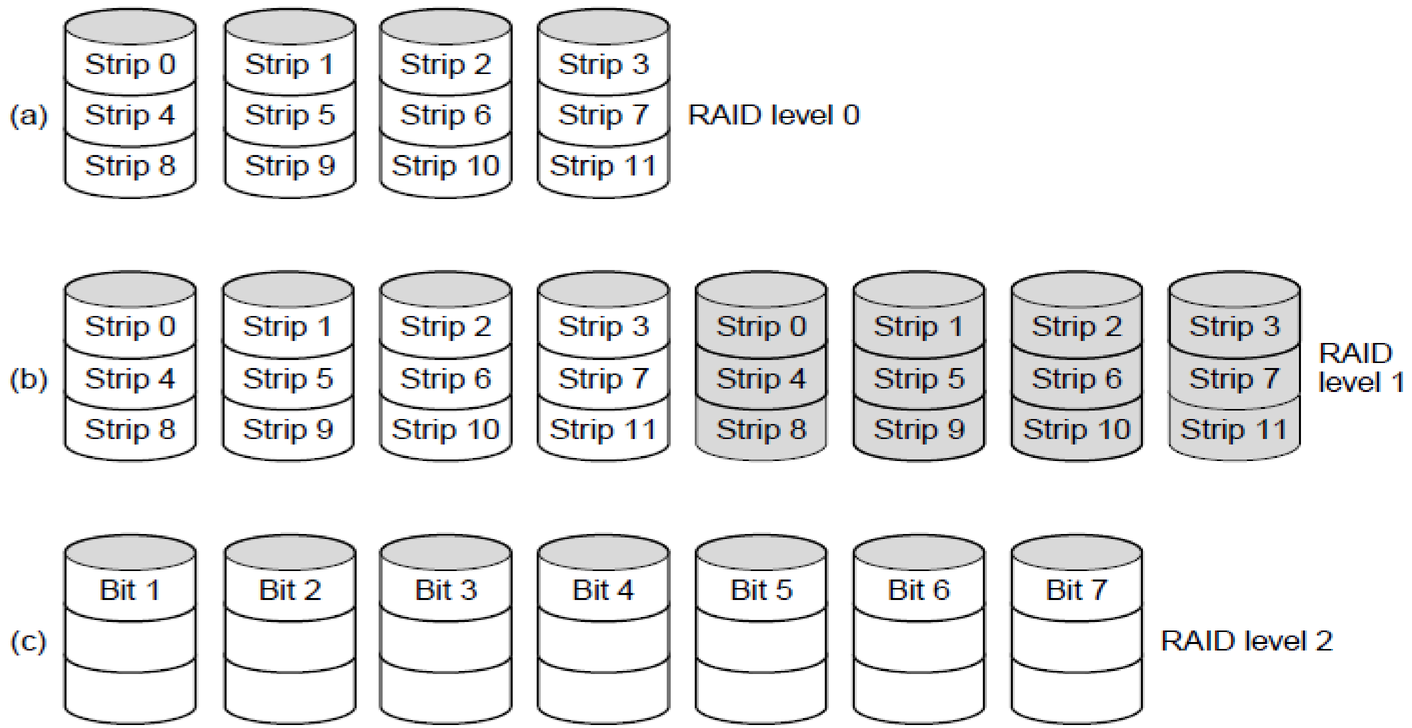
byte	parity
10110011	1
01101010	0

- If a single bit is bad, it is possible to correct it

Mirroring

- Keep two copies of data on two separate disks
- Gives good error recovery
  - if some data is lost, get it from the other source
- Expensive
  - requires twice as many disks
- Write performance can be slow
  - have to write data to two different spots
- Read performance is enhanced
  - can read data from file in parallel

RAID (1) – Redundant Array of (Inexpensive/Independent) Disks



- Figure 2-23. RAID levels 0 through 5. Backup and parity drives are shown shaded.

RAID Level-0

- Often called striping
- Break a file into blocks of data
- Stripe the blocks across disks in the system
- No replication
  disk = file block % number of disks
  sector = file block / number of disks
- Advantages:
  - Simple to implement
  - No overhead
- Disadvantages:
- Provides no redundancy or error detection
  - important to consider because lots of disks means low Mean Time To Failure (MTTF)
  - If array has *n* disks failure rate is *n* times the failure rate of a single disk

RAID Level-1

- A complete file is stored on 4 primary disk
- 4 secondary disks contain an exact copy of the file (mirroring: two copies of each disk block).
- Provides complete redundancy of data (fault-tolerant)
- Simple to implement
- Read performance can be improved
  - file data can be read in parallel
- Write performance suffers
  - must write the data out twice
- Most expensive RAID implementation
  - requires twice as much storage space

RAID Level-2

- Stripes data across disks similar to Level-0
  - difference is data is bit interleaved instead of block interleaved
  - Splitting each byte of the single virtual disk into a pair of 4-bit nibbles, then adding a Hamming code to each one to form a 7-bit word, of which 1, 2, and 4 were parity bits
  - Write the 7-bit Hamming code word over the seven drives, one bit per drive
  - Instead of duplicating the data blocks we use an error correction code
- Uses ECC to monitor correctness of information on disk
- Multiple disks record the ECC information to determine which disk is in fault
- A parity disk is then used to reconstruct corrupted or lost data

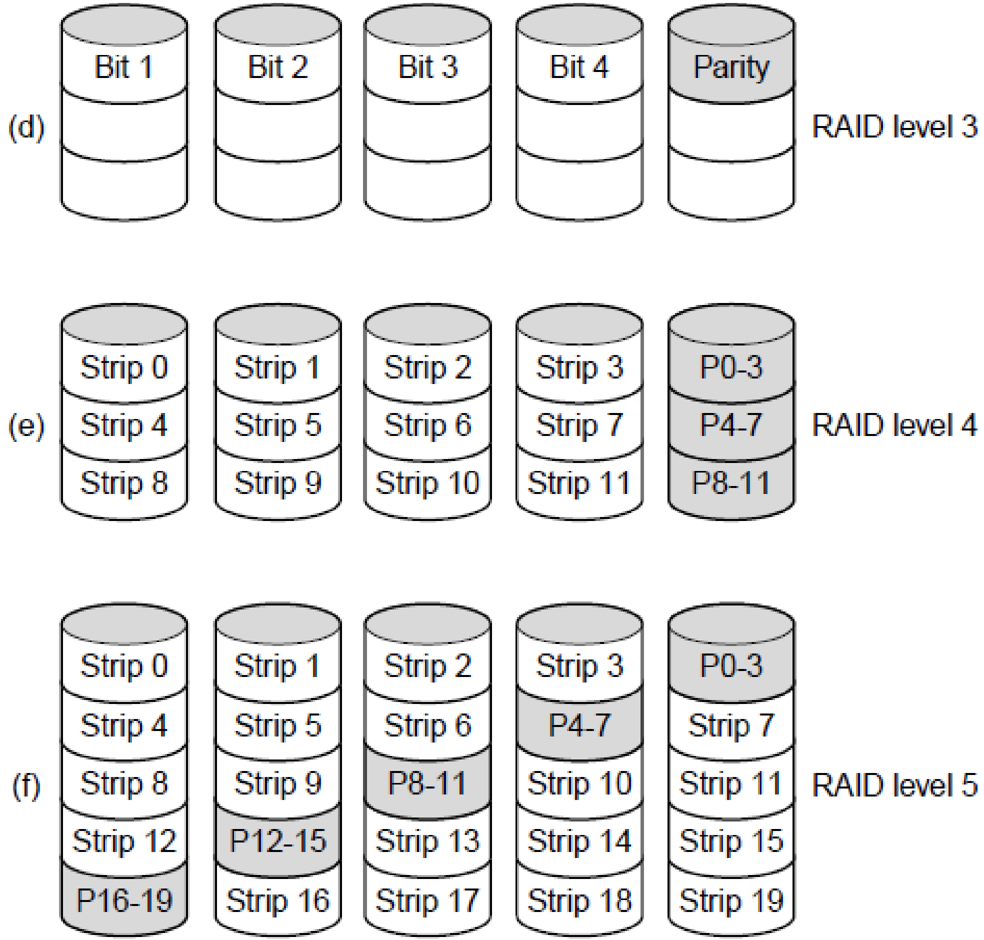
RAID Level-2

- Reconstructing data
  - assume data striped across eight disks
  - correct data: 10011010
  - parity: 0
  - data read: 10011110
  - if we can determine that disk 2 is in error
  - just use read data and parity to know which bit to flip

RAID Level-2

- Requires fewer disks than Level-1 to provide redundancy
- Still needs quite a few more disks
  - for 10 data disks need 4 check disks plus parity disk
- Big problem is performance
  - must read data plus ECC code from other disks
  - for a write, have to modify data, ECC, and parity disks
- Another big problem is only one read at a time
  - while a read of a single block can be done in parallel
  - multiple blocks from multiple files can’t be read because of the bit-interleaved placement of data

RAID (2)



- Figure 2-23. RAID levels 0 through 5. Backup and parity drives are shown shaded.

RAID Level-3

- One big problem with Level-2 is the disks needed to detect which disk had an error
- Modern disks can already determine if there is an error
  - using ECC codes with each sector
- So just need to include a parity disk
  - if a sector is bad, the disk itself tells us, and use the parity disk to correct it
- Requires N+1 disk drives
  - N drives contain data (1/N of each data block)
    - Block b[k] now partitioned into N fragments b[k,1], b[k,2], ... b[k,N]
  - Parity drive contains exclusive or of these N fragments

$$p[k] = b[k,1] \oplus b[k,2] \oplus \ldots \oplus b[k,N]$$

How parity works?

- Truth table for XOR (same as parity)

A	B	A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

Recovering from a disk failure

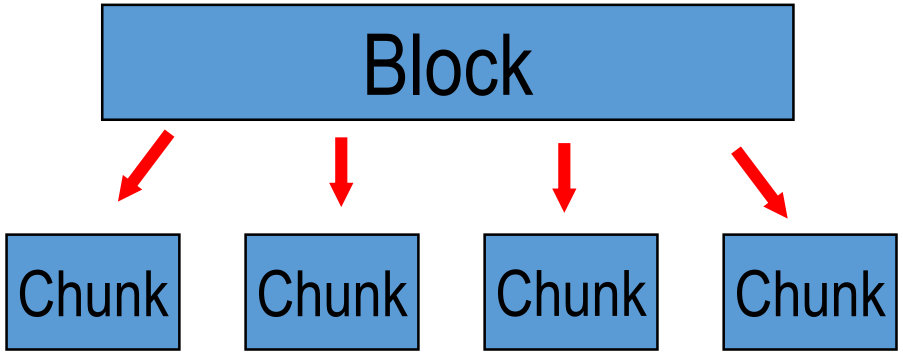
- Small RAID level 3 array with data disks D0 and D1 and parity disk P can tolerate failure of either D0 or D1

D0	D1	P
0	0	0
0	1	1
1	0	1
1	1	0

$D1 \oplus P = D0$	$D0 \oplus P = D1$
0	0
0	1
1	0
1	1

How RAID level 3 works (I)

- Assume we have *N* + 1 disks
- Each block is partitioned into *N* equal chunks

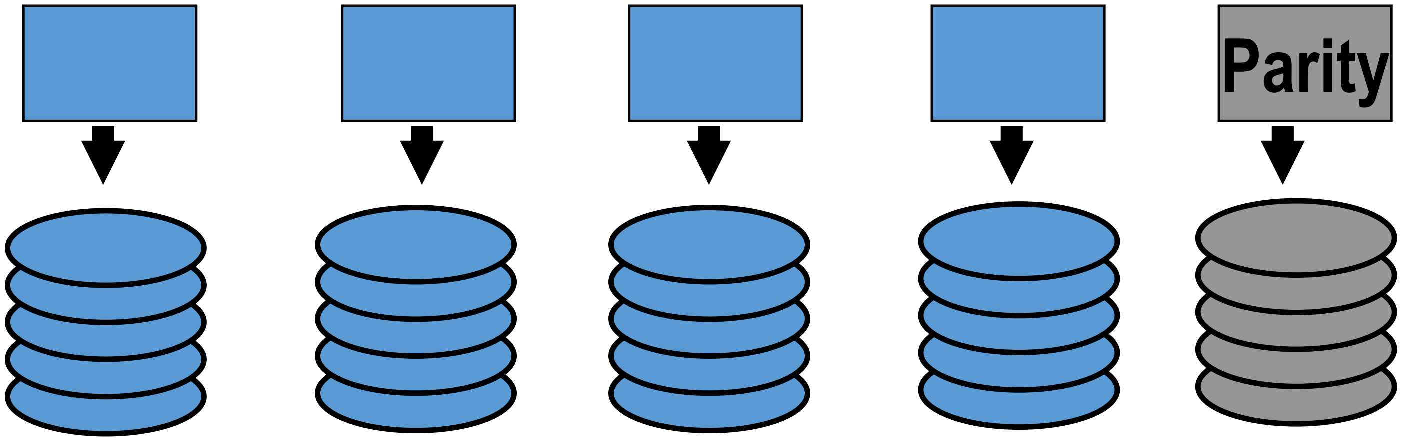


N = 4 in example

How RAID level 3 works (II)

- XOR data chunks to compute the parity chunk

- Each chunk is written into a ***separate disk***

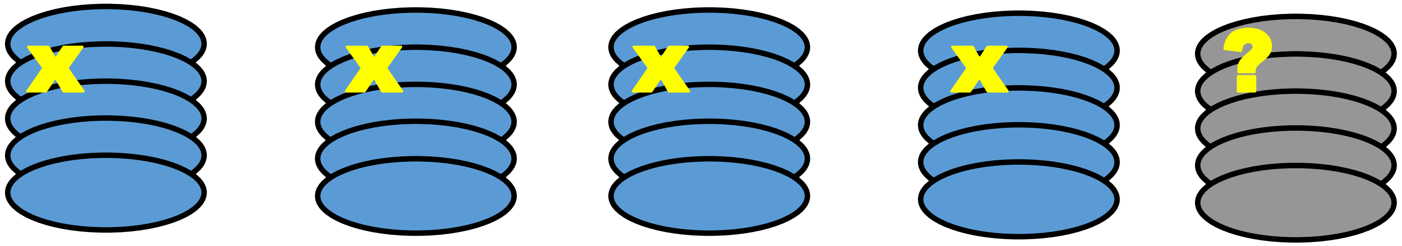


How RAID level 3 works (III)

- Each read/write involves ***all disks*** in RAID array
  - Cannot do two or more reads/writes ***in parallel***
  - Performance of array not better than that of a ***single disk***

RAID LEVEL 4 (I)

- Requires N+1 disk drives
  - N drives contain data
    - Individual blocks, not chunks
  - Blocks with same disk address form a ***stripe***



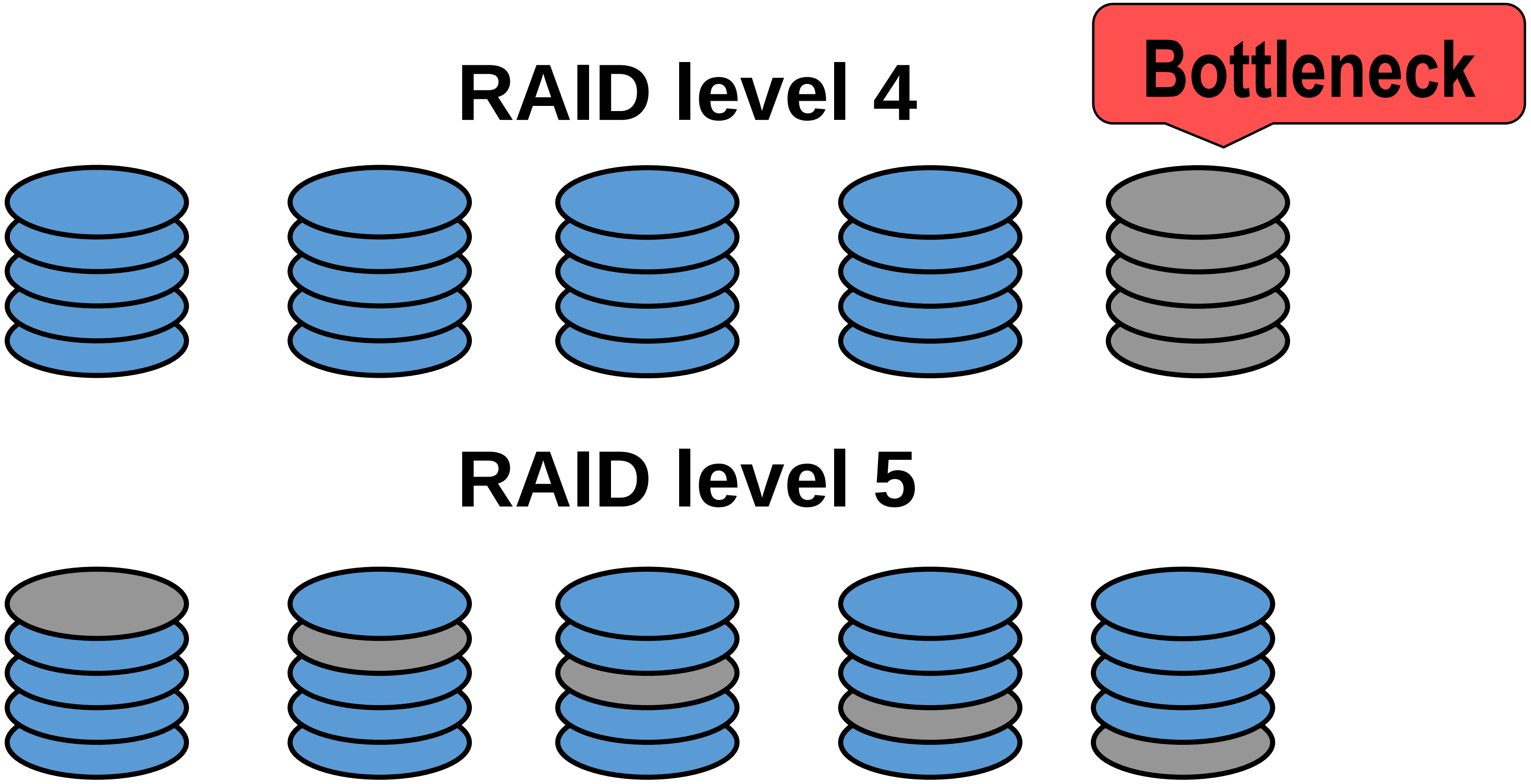
RAID LEVEL 4 (II)

- Parity drive contains **exclusive or** of the *N* blocks in stripe

$$p[k] = b[k] \oplus b[k+1] \oplus \ldots \oplus b[k+N-1]$$

- Parity block now reflects contents of several blocks!
- Can now do parallel reads/writes

RAID levels 4 and 5



RAID Level-4

- Big problem with Level-2 and Level-3 is the bit interleaving
  - To access a single file block of data, must access all the disks
  - Allows good parallelism for a single access but doesn't allow multiple I/O's
- Level-4 interleaves file blocks
  - Allows multiple small I/O's to be done at once

RAID Level-4

- Still use a single disk for parity
- Now the parity is calculated over data from multiple blocks
  - Level-2,3 calculate it over a single block
- If an error detected, need to read other blocks on other disks to reconstruct data

RAID Level-4

- Reads are simple to understand
  - Want to read block A, read it from disk 0
  - If there is an error, read in blocks B,C, D, and parity block and calculate correct data
- What about writes?
  - It looks like a write still requires access to 4 data disks to recalculate the parity data
  - Not true, can use the following formula
    - new parity = (old data xor new data) xor old parity
  - A write requires 2 reads and 2 writes

RAID Level-4

- Doing multiple small reads is now faster than before
- However, writes are still very slow
  - this is because of calculating and writing the parity blocks
- Also, only one write is allowed at a time
  - all writes must access the check disk so other writes have to wait

RAID LEVEL 5

- Single parity drive of RAID level 4 is involved in every write
    - ***Will limit parallelism***
- RAID-5 distribute the parity blocks among the N+1 drives
    - ***Much better***

The small write problem

- Specific to RAID 5
- Happens when we want to update a single block
  - Block belongs to a stripe
  - How can we compute the new value of the parity block

First solution

- Read values of N-1 other blocks in stripe
- Recompute

$$p[k] = b[k] \oplus b[k+1] \oplus \ldots \oplus b[k+N-1]$$

- Solution requires
  - N-1 reads
  - 2 writes (new block and new parity block)

Second solution

- Assume we want to update block b[m]
- Read old values of b[m] and parity block p[k]
- Compute

$$p[k] = \textbf{new}\ b[m] \oplus \textbf{old}\ b[m] \oplus \textbf{old}\ p[k]$$

- Solution requires
  - 2 reads (old values of block and parity block)
  - 2 writes (new block and new parity block)

RAID Level-5

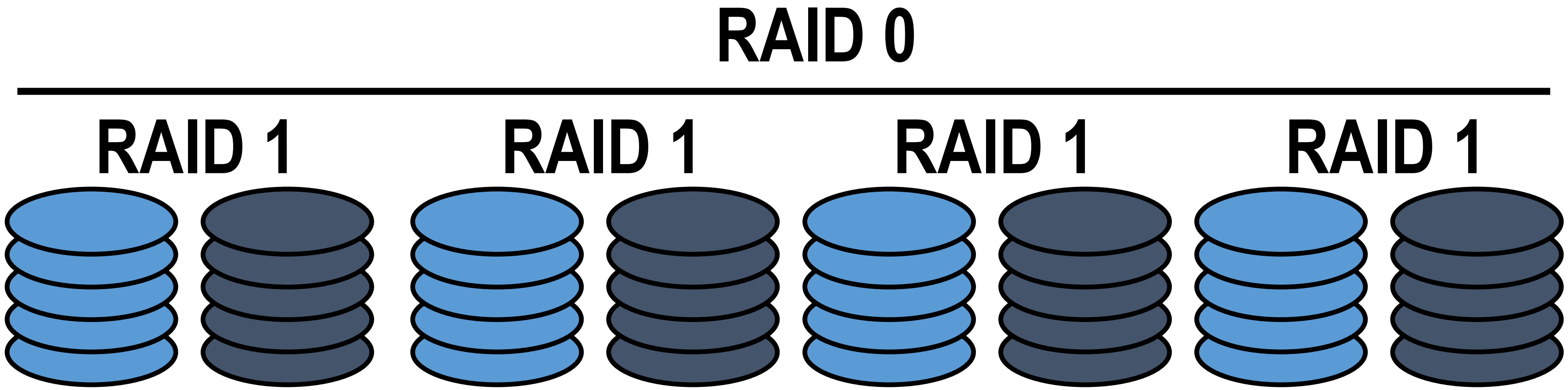
- Level-5 stripes file data and check data over all the disks
  - No longer a single check disk
  - No more write bottleneck
- Drastically improves the performance of multiple writes
  - They can now be done in parallel
- Slightly improves reads
  - One more disk to use for reading

RAID Level-5

- Notice that for Level-4 a write to sector 0 on disk 2 and sector 1 on disk 3 both require a write to disk five for check information
- In Level-5, a write to sector 0 on disk 2 and sector 1 on disk 3 require writes to different disks for check information (disks 5 and 4, respectively)
- Best of all worlds
  - read and write performance close to that of RAID Level-1
  - requires as much disk space as Levels-3,4

Other RAID organizations

- **RAID 10**:
  - Also known as **RAID 1 + 0**
  - Data are striped (as in RAID 0 or RAID 5) over pairs of mirrored disks (RAID 1)



RAID Level-10

- Combine Level-0 and Level-1
- Stripe a files data across multiple disks
  - gives great read/write performance
- Mirror each strip onto a second disk
  - gives the best redundancy
- The most high performance system
- The most expensive system

RAID

- RAID looks like SLED (Simple Large Expensive Disk) to the operating system but have better performance and reliability.
- All RAIDs have the property that the data are distributed over the drives to allow parallel operation.
- There are six different organizations/schemes, each with a different mix of reliability and performance characteristic – known as RAID level 0 through RAID level 5 (defined by Patterson et al.).

CONCLUSION (I)

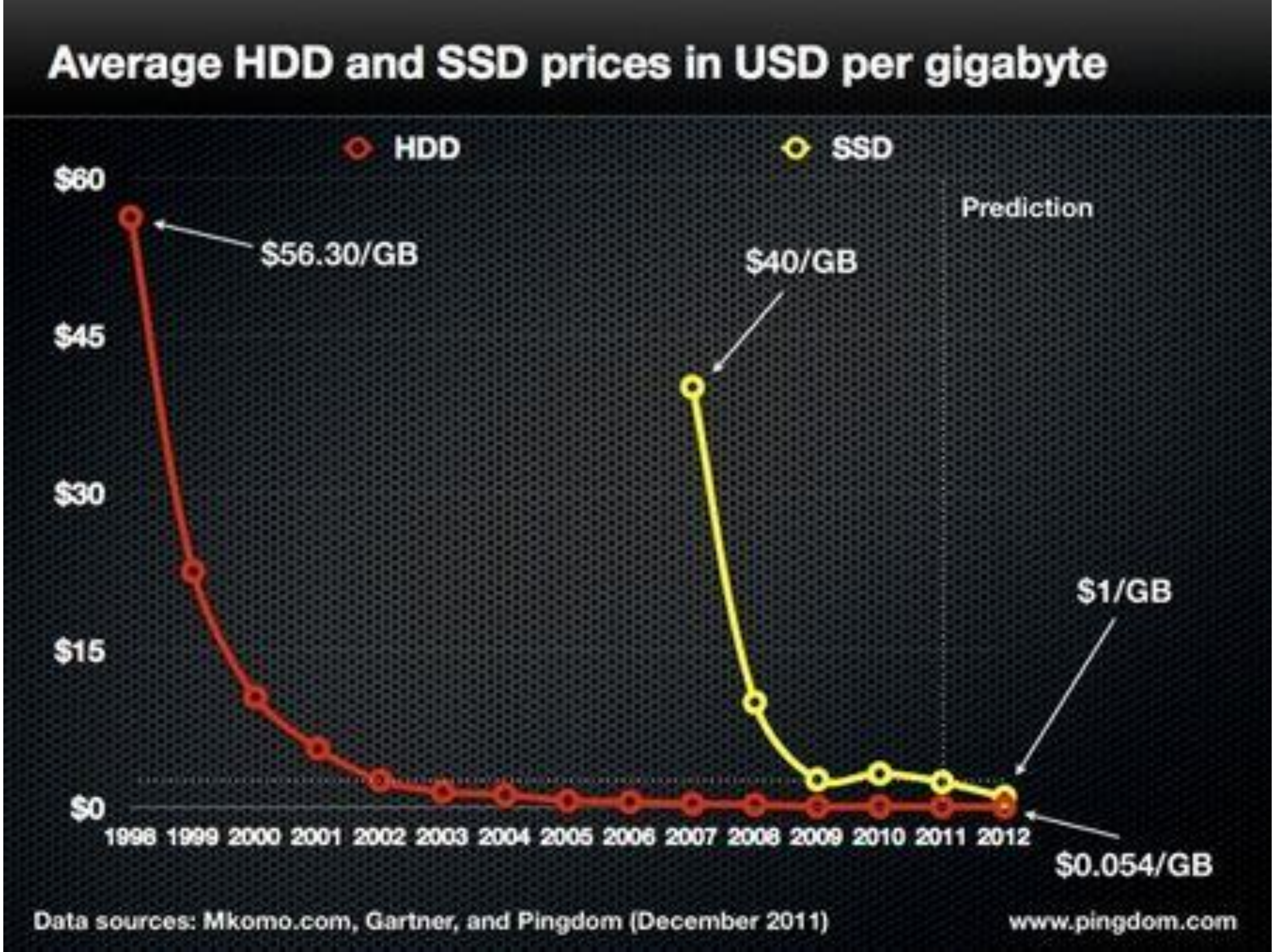
- Original purpose of RAID ***was*** to take advantage of Winchester drives that were smaller and cheaper than conventional disk drives
  - Replace a single drive by an array of smaller drives
- ***Current purpose*** is to build fault-tolerant file systems that do not need backups
- Low cost of disk drives made RAID level 1 attractive for small installations

Beyond Spinning Disks

- Hard drives have been around since 1956
  - The cheapest way to store large amounts of data
  - Sizes are still increasing rapidly
- However, hard drives are typically the slowest component in most computers
  - CPU and RAM operate at GHz
  - PCI-X and Ethernet are GB/s
- Hard drives are not suitable for mobile devices
  - Fragile mechanical components can break
  - The disk motor is extremely power hungry

Advantages of SSDs

- More resilient against physical damage
  - No sensitive read head or moving parts
  - Immune to changes in temperature
- Greatly reduced power consumption
  - No mechanical, moving parts
- Much faster than hard drives
  - >500 MB/s vs ~200 MB/s for hard drives
  - No penalty for random access
    - Each flash cell can be addressed directly
    - No need to rotate or seek
  - Extremely high throughput
    - Although each flash chip is slow, they are RAIDed



Average HDD and SSD prices in USD per gigabyte
HDD
SSD
$60
$45
$30
$15
$0
Prediction
$56.30/GB
$40/GB
$1/GB
$0.054/GB
1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012
Data sources: Mkomo.com, Gartner, and Pingdom (December 2011)
www.pingdom.com

Challenges with Flash

- Flash memory is written in pages, but erased in blocks
  - Pages: 4 – 16 KB, Blocks: 128 – 256 KB
  - Thus, flash memory can become fragmented
  - Leads to the write amplification problem
- Flash memory can only be written a fixed number of times
  - Typically 3000 – 5000 cycles for MLC
  - SSDs use wear leveling to evenly distribute writes across all flash cells