Mass-Storage Structure

Operating Systems

Wenbo Shen

Acknowledgement: some pages are based on the slides from Yajin Zhou@ZJU and Henri Casanova@Hawaii.

Content

- Overview of Mass Storage Structure
- Disk Structure
- Disk Scheduling
- Disk Management
- Swap-Space Management
- RAID Structure

Overview

- Magnetic disks provide bulk of secondary storage of computer system
  - **hard disk** is most popular; some magnetic disks could be removable
  - driver attached to computer via I/O buses (e.g., USB, SCSI, EIDE, SATA…)
  - drives rotate at 60 to 250 times per second (7200rpm = **120rps**)
- Magnetic disks has platters, range from .85” to 14” (historically)
  - 3.5”, 2.5”, and 1.8” are common nowadays
- Capacity ranges from 30GB to 3TB per drive, and even bigger

The First Commercial Disk Drive

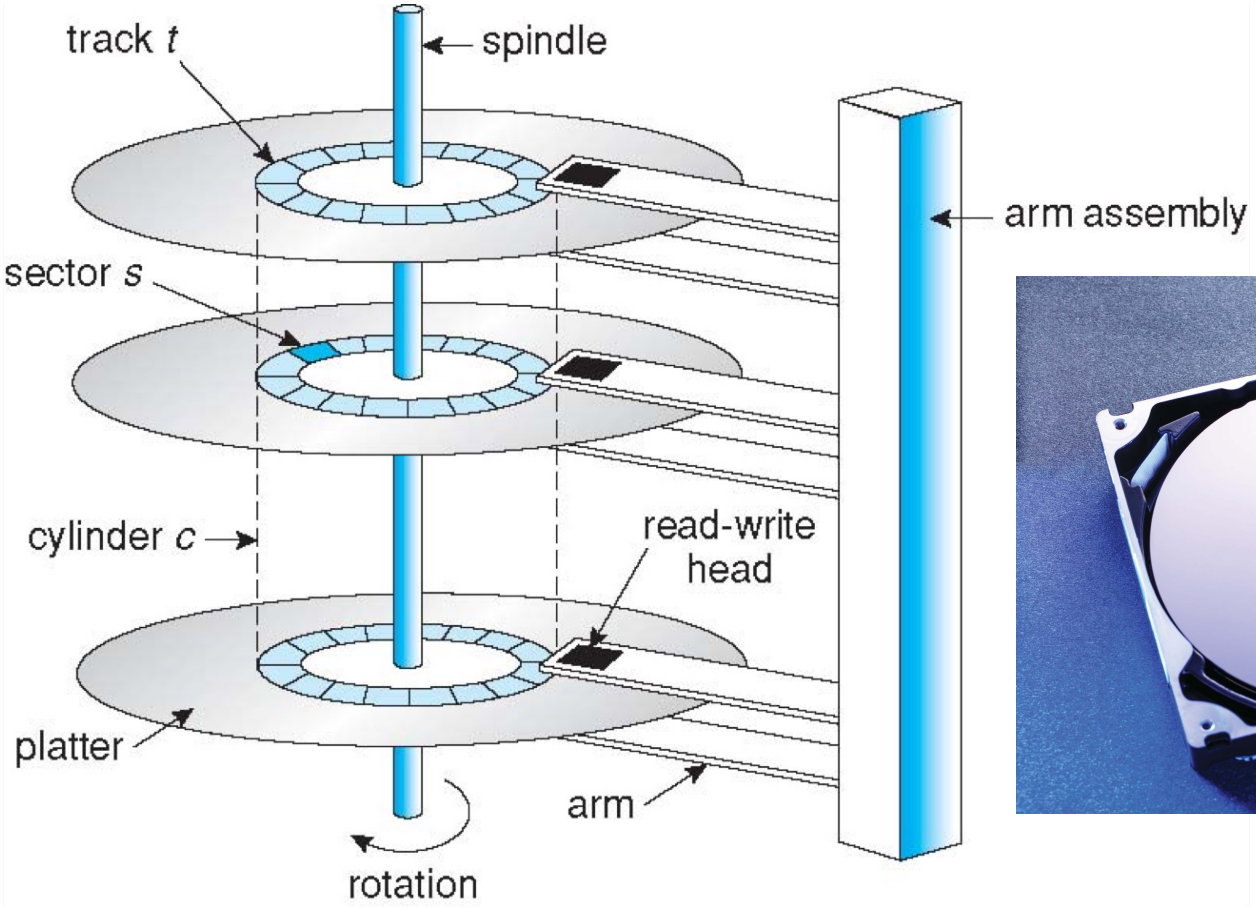
1956 IBM RAMDAC computer included the IBM Model 350 disk storage system

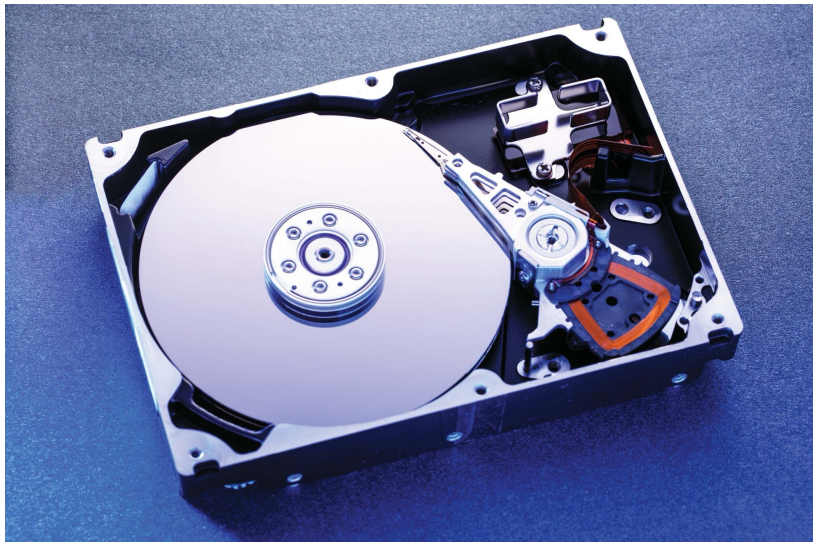
5M (7 bit) characters
50 x 24” platters
Access time = < 1 second

Disk Structure

- Disk drives are addressed as a 1-dimensional arrays of logical blocks (LBA)
  - logical block is the smallest unit of transfer
- Logical blocks are mapped into **sectors** of the disk sequentially
  - sector 0 is the first sector of the first track on the outermost cylinder
  - mapping proceeds in order
    - first through that **track**
    - then the rest of the tracks in that **cylinder**
    - then through the rest of the cylinders from outermost to innermost
  - logical to physical address should be easy
    - except for bad sectors

Moving-head Magnetic Disk





Magnetic Disk

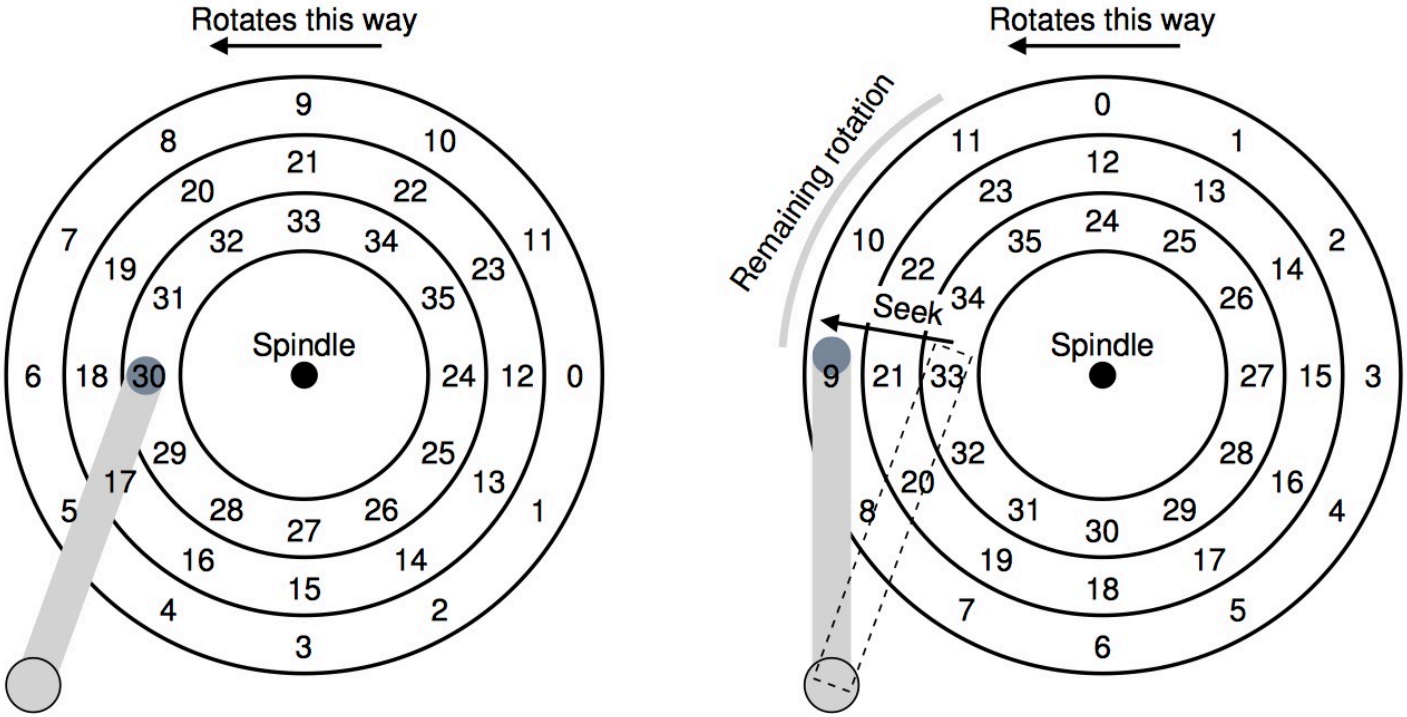


Figure 37.3: **Three Tracks Plus A Head (Right: With Seek)**

Magnetic Disk

- **Positioning time** is time to move disk arm to desired sector
  - positioning time includes **seek time** and **rotational latency**
    - seek time: move disk to the target cylinder
    - rotational latency: for the target sector to rotate under the disk head
  - positioning time is also called random-access time
- **Performance**
  - **transfer rate**: **theoretical** 6 Gb/sec; **effective** (real) about 1Gb/sec
    - Transfer rate is rate at which data flow between drive and computer
  - **seek time** from 3ms to 12ms (9ms common for desktop drives)
  - latency based on spindle speed: 1/rpm * 60
    - average latency = ½ latency

Spindle [rpm]	Average latency [ms]
4200	7.14
5400	5.56
7200	4.17
10000	3
15000	2

Magnetic Disk

- **Average access time** = average seek time + average latency
  - for fastest disk 3ms + 2ms = 5ms;
  - for slow disk 9ms + 5.56ms = 14.56ms
- **Average I/O time**: average access time + (data to transfer / transfer rate) + controller overhead
  - e.g., to transfer a 4KB block on a 7200 RPM disk; 5ms average seek time, 1Gb/sec transfer rate with a .1ms controller overhead:

5ms + 4.17ms + 4KB / 1Gb/sec + 0.1ms (4.17 is average latency)

Disk Scheduling

- OS is responsible for using hardware efficiently
  - for the disk drives: a fast access time and high disk bandwidth
  - **access time**: seek time (roughly linear to seek distance) + rotational latency
  - **disk bandwidth** is the speed of data transfer, data /time
    - data: total number of bytes transferred
    - time: between the first request and completion of the last transfer

Disk Scheduling

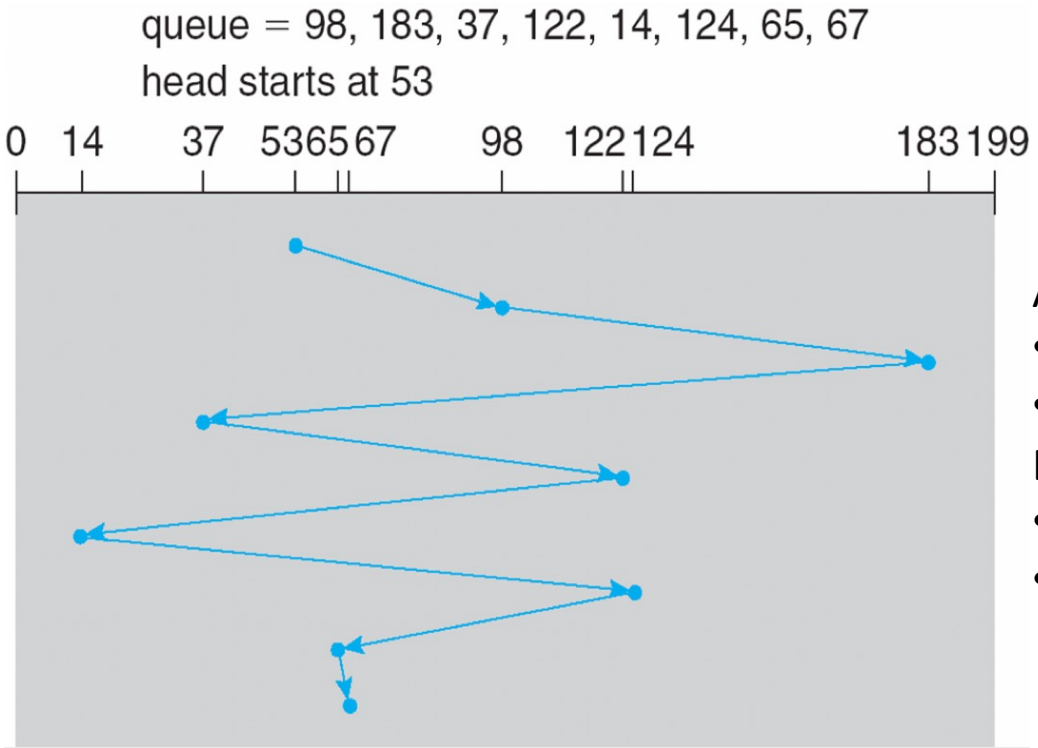
- **Disk scheduling** chooses which **pending disk request to service next**
  - concurrent sources of disk I/O requests include OS, system/user processes
  - idle disk can immediately work on a request, otherwise os queues requests
    - each request provide I/O mode, disk & memory address, and # of sectors
    - OS maintains a queue of requests, per disk or device
    - optimization algorithms only make sense when a queue exists
  - In the past, operating system is responsible for queue management, disk drive head scheduling
    - Now, **built into the storage devices, controllers - firmware**
    - Just provide **LBA(**Logical block addressing**)**, handle sorting of requests
      - Some of the algorithms they use described next

Disk Scheduling

- Disk scheduling usually tries to minimize **seek time**
  - rotational latency is difficult for OS to calculate
- There are many disk scheduling algorithms
  - FCFS
  - SSTF
  - SCAN, C-SCAN
  - LOOK, C-LOOK
- We use a request queue of cylinders "**98, 183, 37, 122, 14, 124, 65, 67**" (**[0, 199]**), and initial head position **53** as the example

FCFS

- First-come first-served, simplest scheduling algorithm
- Total head movements of *640* cylinders



Advantage:

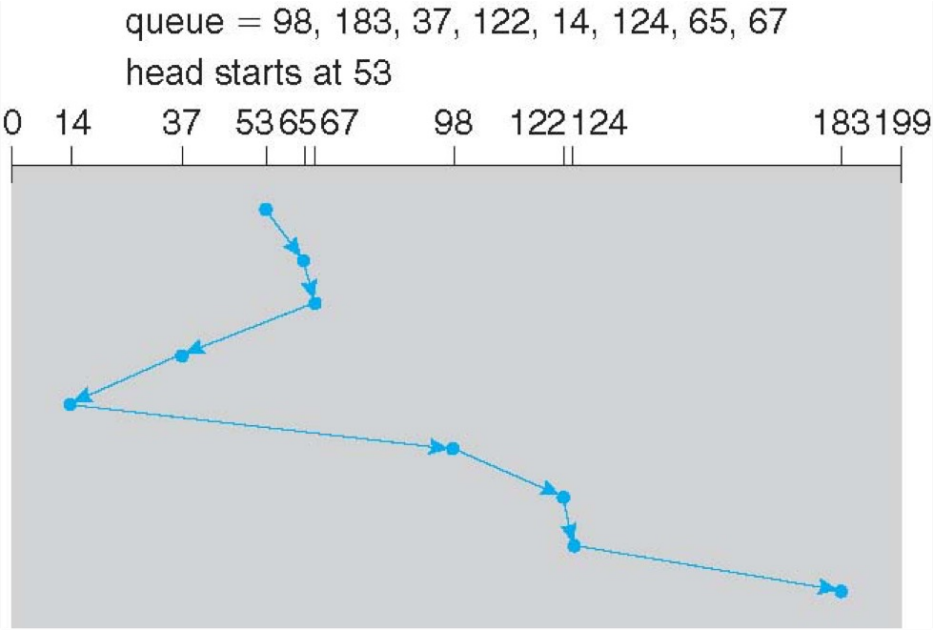
- Every request gets a fair chance
- No indefinite postponement

Disadvantages:

- Does not try to optimize seek time
- May not provide the best possible service

SSTF

- SSTF: shortest seek time first
  - selects the request with minimum seek time from the **current** head position
  - SSTF scheduling is a form of SJF scheduling, **starvation** may exist
    - unlike SJF, SSTF **may not** be **optimal**
- Total head movement of *236* cylinders



Advantage:

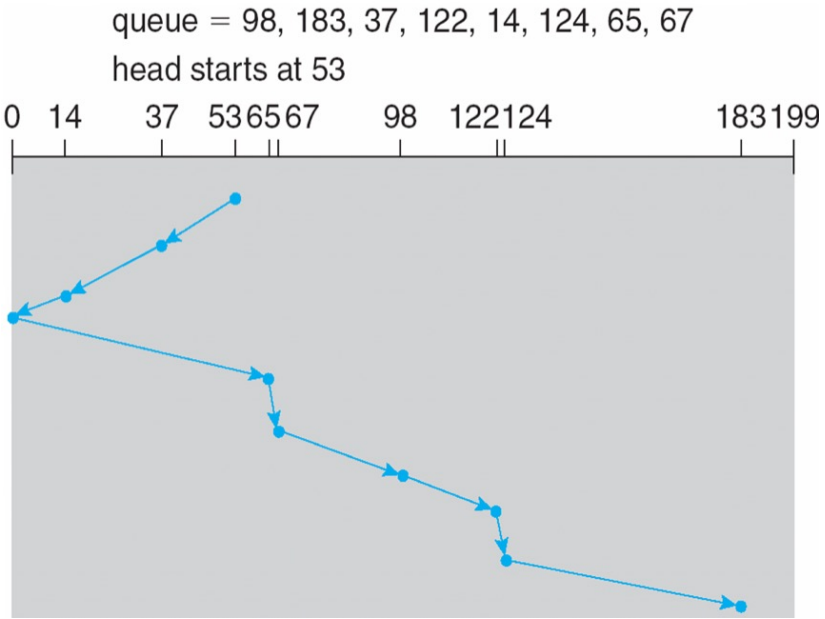
- Average Response Time decreases
- Throughput increases

Disadvantages:

- Overhead to calculate seek time in advance
- Can cause Starvation for a request if it has higher seek time as compared to incoming requests
- High variance of response time as SSTF favors only some requests

SCAN

- SCAN algorithm sometimes is called the **elevator** algorithm
  - disk arm starts at one **end** of the disk, and moves toward the **other end**
  - service requests during the movement until it gets to the other end
  - then, the head movement is reversed and servicing continues.

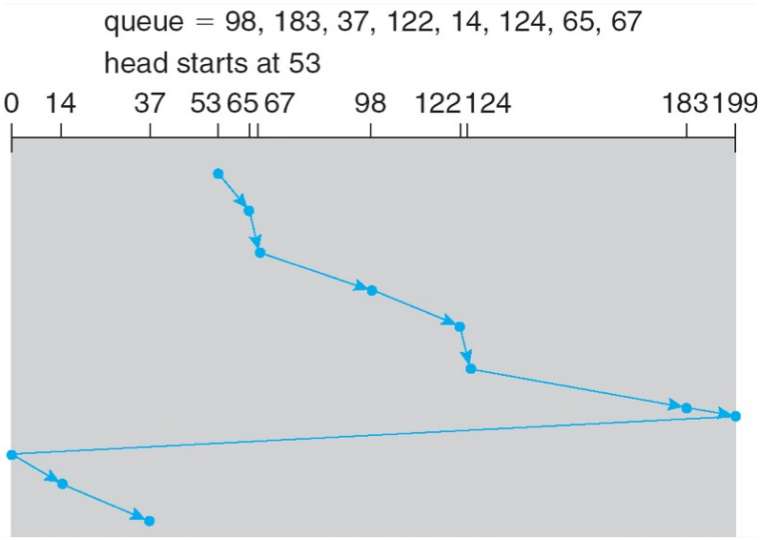


Advantage:
- High throughput
- Low variance of response time
- Average response time

Disadvantages:
- Long waiting time for requests for locations just visited by disk arm

C-SCAN

- Circular-SCAN is designed to provides a more uniform wait time
  - head moves from **one end** to **the other**, servicing requests while going
  - when the head reaches the end, it immediately returns to the beginning
    - **without** servicing any requests on the return trip
  - it essentially treats the cylinders as a circular list

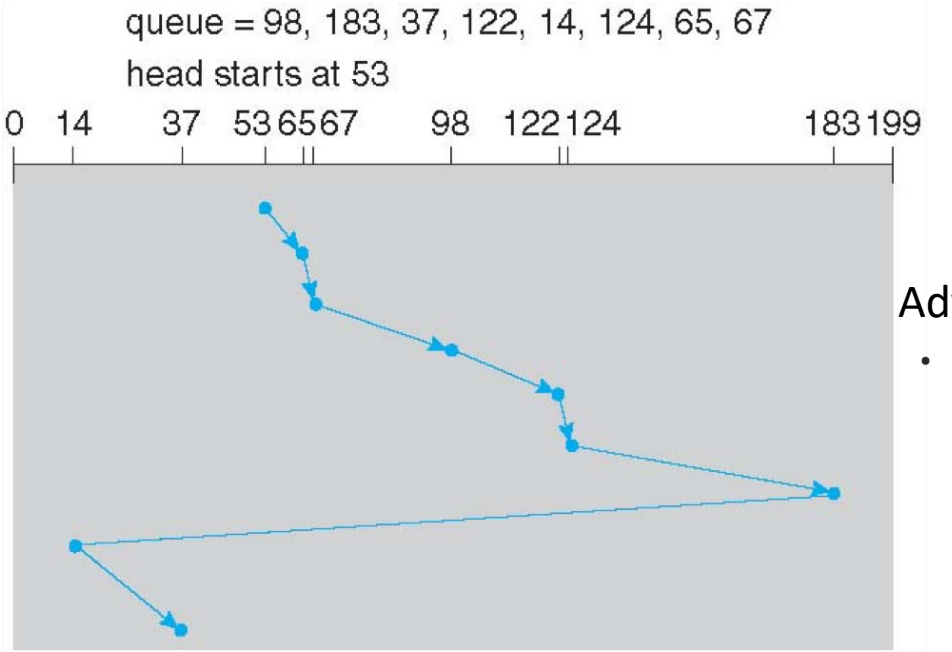


Advantage:

- Provides more uniform wait time compared to SCAN

LOOK/C-LOOK

- SCAN and C-SCAN moves head end to end, even no I/O in between
  - in implementation, head only goes as far as **last request** in each direction
  - **LOOK** is a version of **SCAN**, **C-LOOK** is a version of **C-SCAN**



Advantage:

- prevents the extra delay which occurred due to unnecessary traversal to the end of the disk.

Selecting Disk-Scheduling Algorithm

- Disk scheduling performance depends on the # and types of requests
  - disk-scheduling should be written as a separate, replaceable, module
    - SSTF is common and is a reasonable choice for the default algorithm
    - LOOK and C-LOOK perform better for systems that have heavy I/O load
  - disk performance can be influenced by file-allocation and metadata layout
    - file systems spend great deal of efforts to increase spatial locality

Nonvolatile Memory Devices

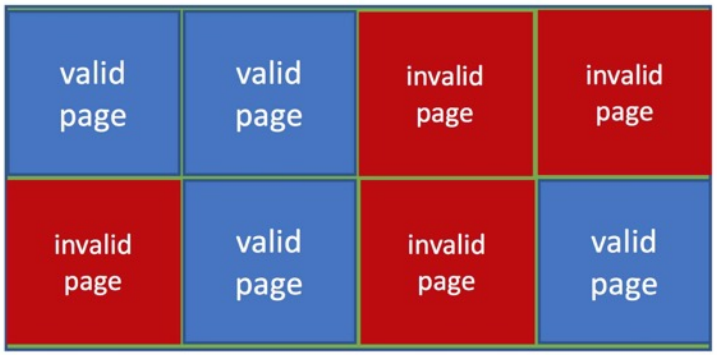
- If disk-drive like, then called solid-state disks (SSDs)
- Other forms include **USB drives** (thumb drive, flash drive), DRAM disk replacements, surface-mounted on motherboards, and main
- Can be **more reliable** than HDDs
- More expensive per MB
- Maybe have shorter life span – need careful management
- Less capacity, but much faster
- Busses can be too slow -> connect directly to PCI for example
- No moving parts, so no seek time or rotational latency
  - First come first served is good

Nonvolatile Memory Devices

- Have characteristics that present challenges
  - **Read and written in "page"** increments (think sector) but can't **overwrite** in place
    - Must first be **erased**, and erases happen in larger "**block**"
    - Assume block size: 64k – 16 pages, page size: 4k
    - Can only be erased a limited number of times before worn out – ~ 100,000
    - Life span measured in **drive writes per day (DWPD)**
      - A 1TB NAND drive with rating of 5DWPD is expected to have 5TB per day written within warrantee period without failing

NAND Flash Controller Algorithms

- With no overwrite, pages end up with mix of valid and invalid data
- To track which logical blocks are valid, controller maintains **flash translation layer (FTL) table**
- Also implements **garbage collection** to free invalid page space - pages available but no free blocks
- Allocates over-provisioning to provide working space for GC
  - Copy good data to over-provisioning area, and erase the block for later use
- Each cell has lifespan, so need to write equally to all cells



NAND block with valid and invalid pages

Magnetic Tape

- Tape was early type of secondary storage, now mostly for backup
  - large capacity: 200GB to 1.5 TB
  - slow access time, especially for random access
    - seek time is much higher than disks
    - once data under head, transfer rates comparable to disk (140 MB/s)
    - need to wind/rewind tape for random access
  - data stored on the tape are relatively permanent

Disk Management

- **Physical formatting**: divide disk into sectors for controller to read/write
  - Each sector can hold header information, plus data, plus error correction code (ECC)
  - Usually 512 bytes of data but can be selectable
- OS records its own data structures on the disk
  - **partition disk** into groups of cylinders, each treated as a logical disk
  - **logical formatting** partitions to **make a file system** on it
    - some FS has spare sectors reserved to handle bad blocks
    - FS can further group blocks into clusters to improve performance
      - Disk I/O done in blocks
      - File I/O done in **clusters**
  - initialize the boot sector if the partition contains OS image

Disk Management-Device names

- By convention, IDE(Integrated Drive Electronics) drives will be given device names /dev/hda to /dev/hdd.
  - Hard Drive A (/dev/hda) is the first drive
  - Hard Drive C (/dev/hdc) is the third drive
- SCSI(Small Computer System Interface) drives follow a similar pattern; They are represented by 'sd' instead of 'hd'.
  - Partition starts from 1, such as /dev/sda1 or /dev/hda1
  - **The first partition** of the **second** SCSI drive would therefore be /dev/sdb1.

Disk Management-Partition

- There are at most 4 primary partitions on a disk

drive name	drive controller	drive number	partition type	partition number
/dev/hda1	1	1	primary	1
/dev/hda2	1	1	primary	2
/dev/hda3	1	1	primary	3
/dev/hda4	1	1	swap	NA
/dev/hdb1	1	2	primary	1
/dev/hdb2	1	2	primary	2
/dev/hdb3	1	2	primary	3
/dev/hdb4	1	2	primary	4

- If we need to have more than 4 partitions, we need to use the logical partition - number starts from 5

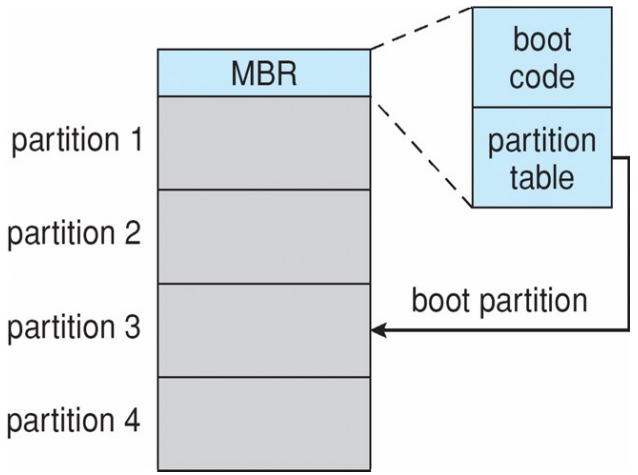
drive name	drive controller	drive number	partition type	partition number
/dev/hdb1	1	2	primary	1
.....				
/dev/hda5	1	2	logical	2
/dev/hdb6	1	2	logical	3

Disk Management

- **Root partition** contains the OS, other partitions can hold other OSes, other file systems, or be raw
  - **Mounted** at boot time
  - Other partitions can mount automatically or manually
- At mount time, file system consistency checked
  - Is all metadata correct?
    - If not, fix it, try again
    - If yes, add to mount table, allow access
- **Boot block** can point to boot volume or boot loader set of blocks that contain enough code to know how to load the kernel from the file system
  - Or a boot management program for multi-os booting

Disk Management

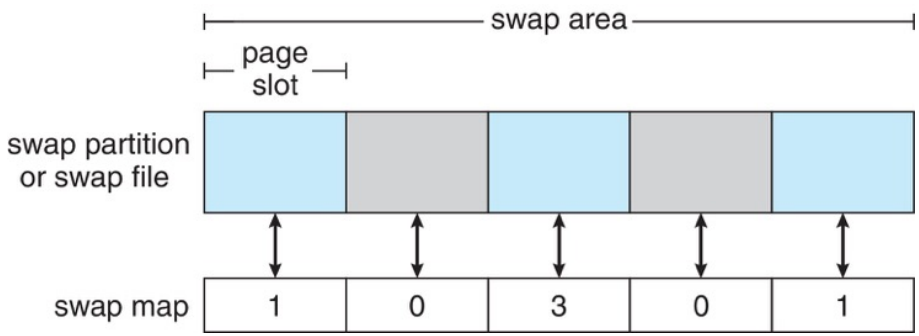
- Raw disk access for apps that want to do their own block management, keep OS out of the way (databases for example)
- Boot block initializes system
  - The bootstrap is stored in ROM, firmware
  - **Bootstrap loader** program stored in boot blocks of boot partition
- Methods such as **sector sparing** used to handle bad blocks



Booting from secondary storage in Windows

Swap Space Management

- Used for moving entire processes (swapping), or pages (paging), from DRAM to secondary storage when DRAM not large enough for all processes
- Operating system provides **swap space management**
  - Secondary storage slower than DRAM, so important to optimize performance
  - Multiple swap spaces possible – decreasing I/O load on any given device
  - Best to have dedicated devices
  - Can be in separate partition or a file within a file system (for convenience of adding)
  - Data structures for swapping on Linux systems:
    - 0 means not used
    - 3 means mapped to 3 proc



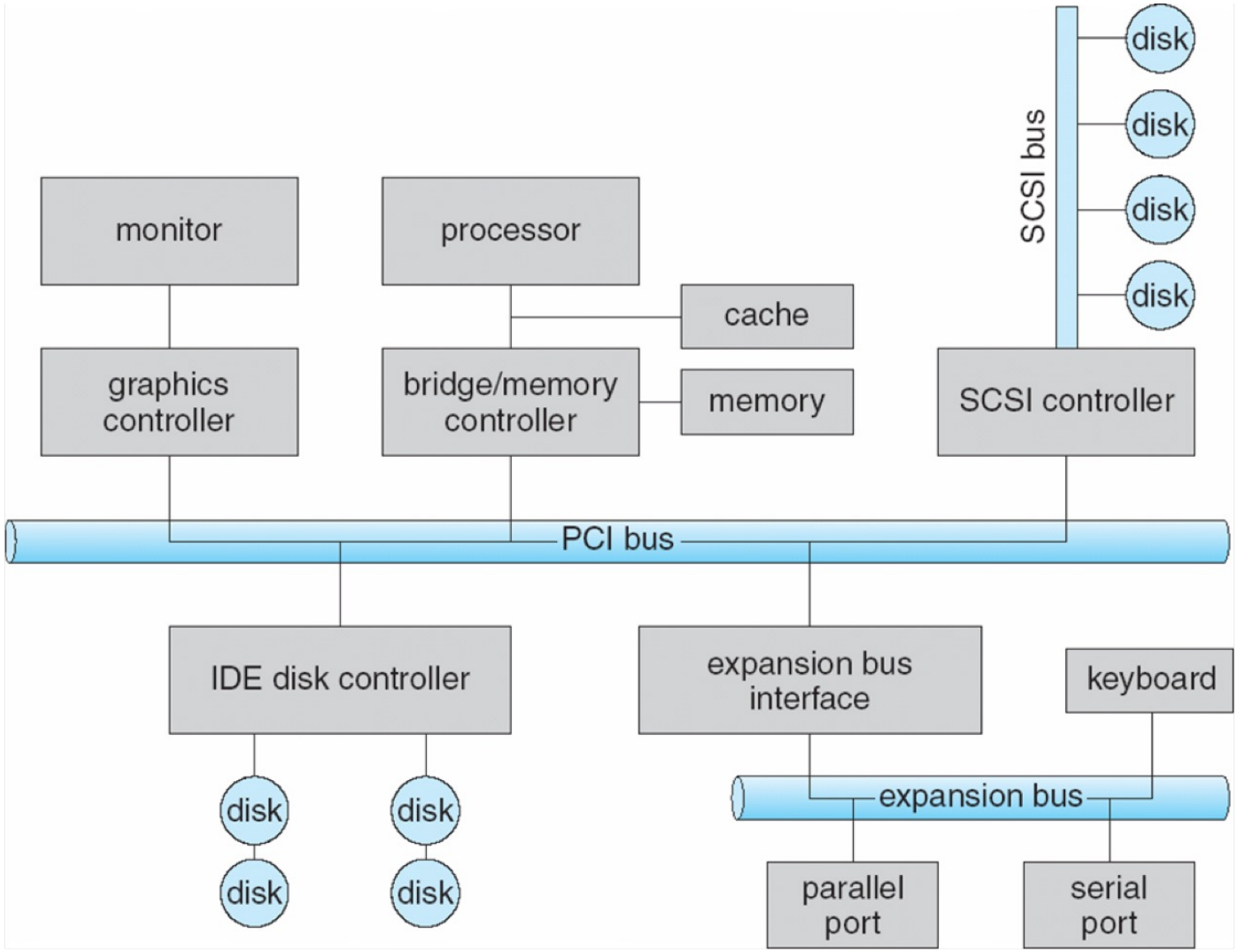
Disk Attachment

- Disks can be attached to the computer as:
  - **host-attached** storage
    - hard disk, RAID arrays, CD, DVD, tape…
  - **network-attached** storage
  - **storage area network**

Host-Attached Storage

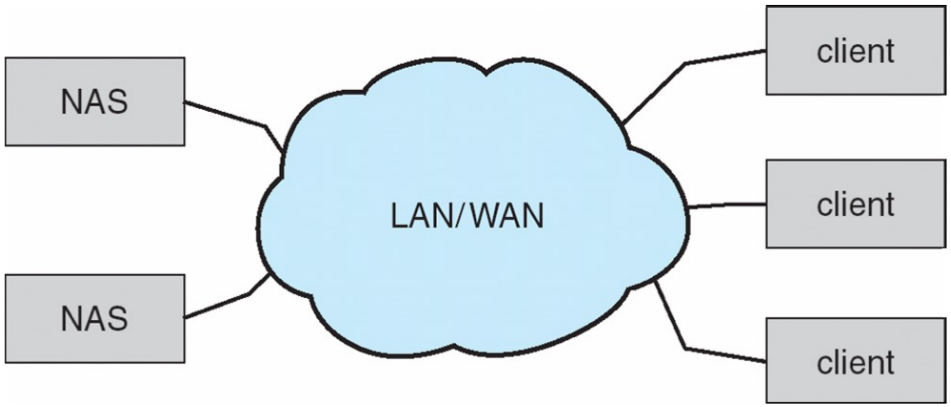
- Disks can be attached to the computers directly via an **I/O bus**
  - e.g., SCSI（Small Computer System Interface）is a bus architecture, up to 16 devices on one cable,
    - SCSI initiator requests operations; SCSI targets(e.g., disk) perform tasks; each target can have up to 8 logical units
    - Linux use /dev/sda represent an SCSI disk drive
      - sd stands for SCSI disk
  - e.g., Integrated Drive Electronics (IDE) interface
    - Linux use /dev/hda represent an IDE disk drive
  - e.g., Fiber Channel is high-speed serial bus
    - can be switched fabric with 24-bit address space
    - most common storage area networks (SANs) interconnection

A Typical PC Bus Structure



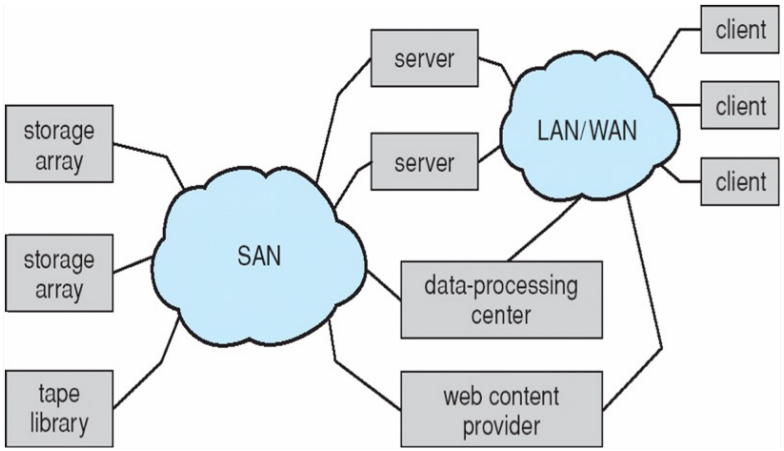
Network-Attached Storage

- **NAS** is storage made available over a network instead of a local bus
  - client can remotely attach to file systems on the server
  - NFS, CIFS, and iSCSI are common protocols
  - usually implemented via remote procedure calls (RPCs)
  - typically over TCP or UDP on IP network
    - **iSCSI** protocol uses IP network to carry the SCSI protocol

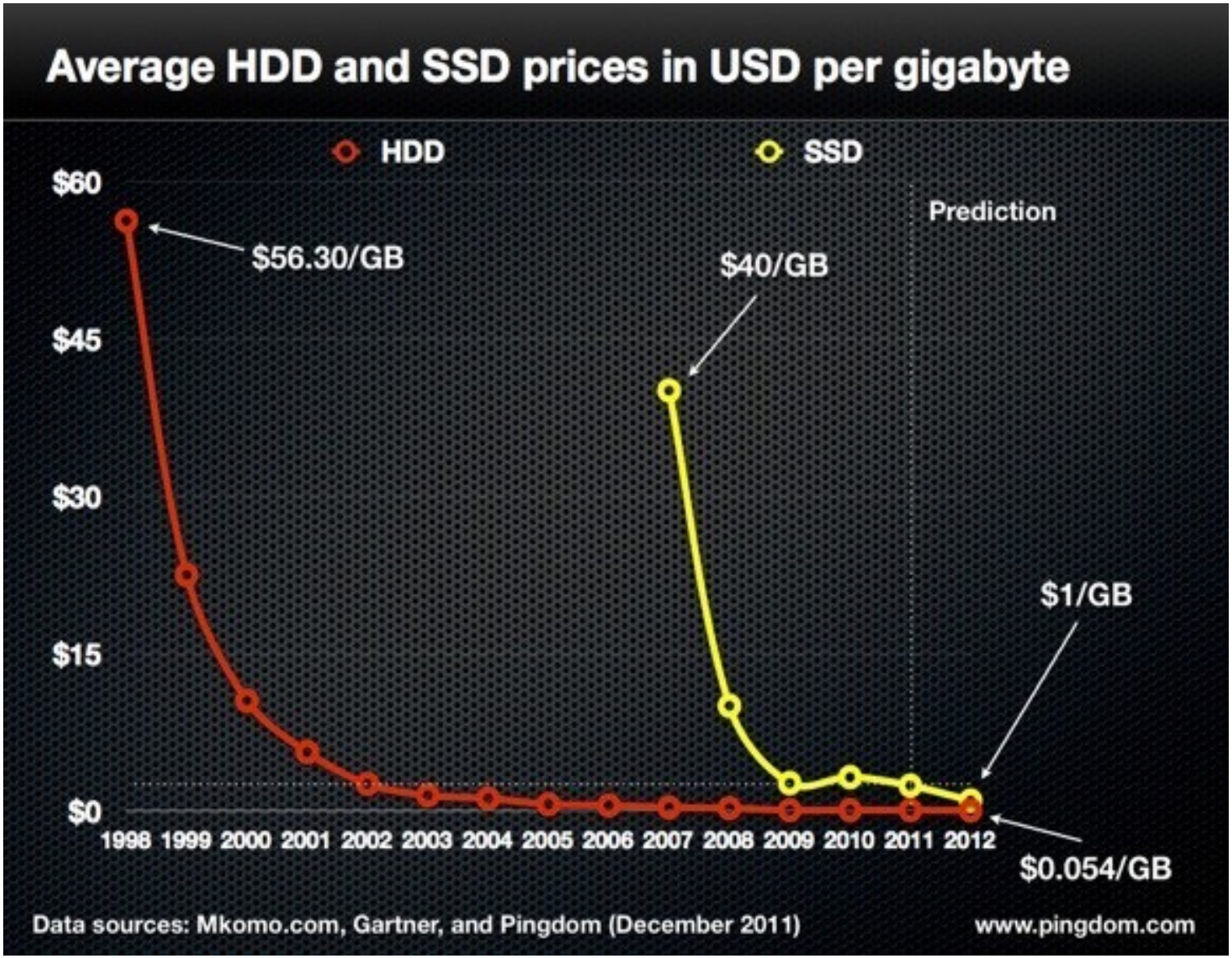


Storage Area Network

- **SAN** is a private network connecting servers and storage units
  - SAN consumes high bandwidth on the data network, separation is needed
  - TCP/IP stack less efficient for storage access
    - SAN uses **high speed interconnection and efficient protocols**
    - FC (Infiniband) is the most common SAN interconnection
  - multiple hosts and storage arrays can attach to the same SAN
    - a *cluster* of servers can share the same storage
  - storage can be *dynamically* allocated to hosts

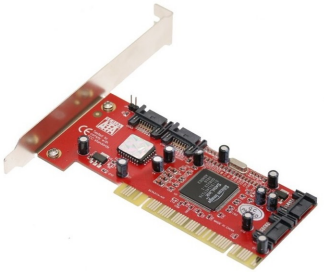


Disks are becoming cheaper



RAID

- Disks are unreliable, slow, but cheap
- Simple idea: let's use redundancy
  - Increases reliability
    - If one fails, you have another one
  - Increases speed
    - Aggregate disk bandwidth if data is split across disks
- Redundant Array of Independent Disks (RAID)
  - The OS can implement it with multiple bus-attached disks
  - A RAID controller in hardware
  - A "RAID array" as a stand-alone box



RAID

- Data Mirroring
  - Keep the same data on multiple disks
    - Every write is to each mirror, which takes time
- Data Striping
  - Keep data split across multiple disks to allow parallel reads
    - e.g., read bits of a byte from 8 disks
- Error-Code Correcting (ECC) - Parity Bits
  - Keep information from which to reconstruct lost bits due to a drive failing
- These techniques are combined at will

RAID Levels

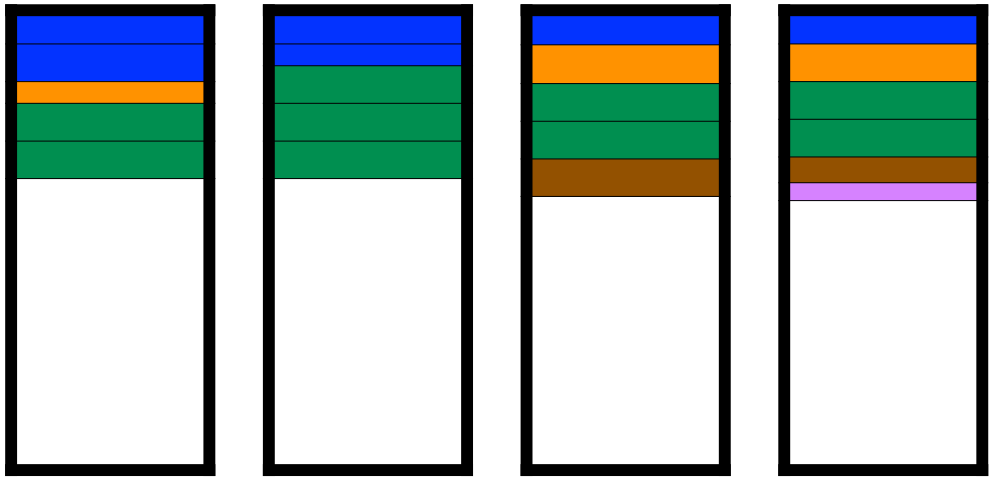
- Combinations of the techniques are called "levels"
  - More of a marketing tool, really
- You should know about common RAID levels, i.e.: 0, 1, 1+0, 5, 5+0, 6, 6+0
  - The book talks about all of them
  - but for level 2, which is not used

RAID 0

- **RAID 0**: splits data evenly across two or more disks without parity bits（No redundancy）
- Data is striped across multiple disks
  - Using a fixed strip size
- Gives the illusion of a larger disk with high bandwidth when reading/writing a file
  - Accessing a single strip is not faster
- Improves performance, but not reliability
- Useful for high-performance

RAID 0 Example

- Fixed strip size
- 5 files of various sizes
- 4 disks

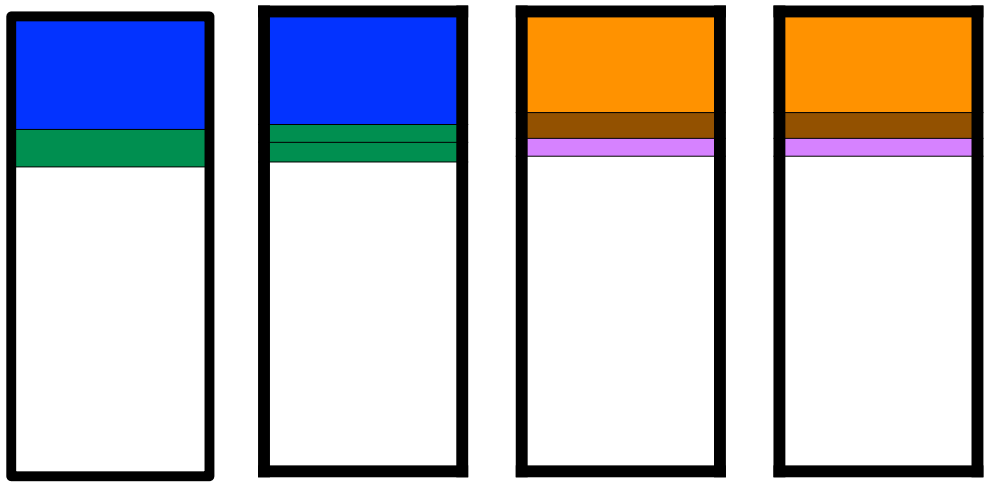


RAID 1

- an exact copy (or mirror) of a set of data on two disks
- Mirroring (also called shadowing)
- Write every written byte to 2 disks
  - Uses twice as many disks as RAID 0
- Reliability is ensured unless you have (extremely unlikely) simultaneous failures
- Performance can be boosted by reading from the disk with the fastest seek time
  - The one with the arm the closest to the target cylinder

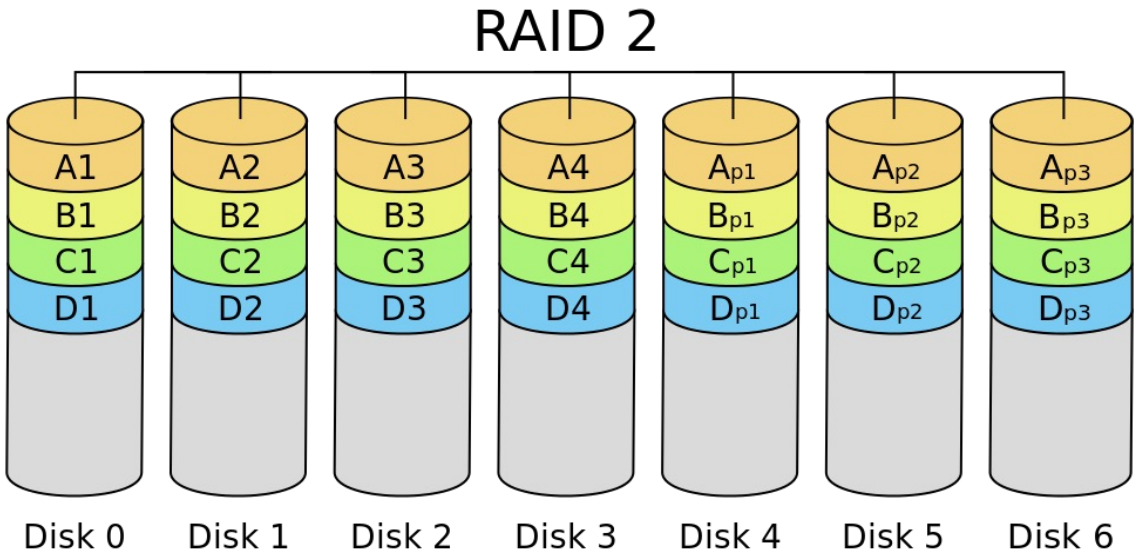
RAID 1 Example

- 5 files of various sizes
- 4 disks



RAID2

- **RAID 2**: stripes data at the **bit**-level; uses Hamming code for error correction (not used)
  - hamming code (4bit data+3bit parity) allows 7 disks to be used

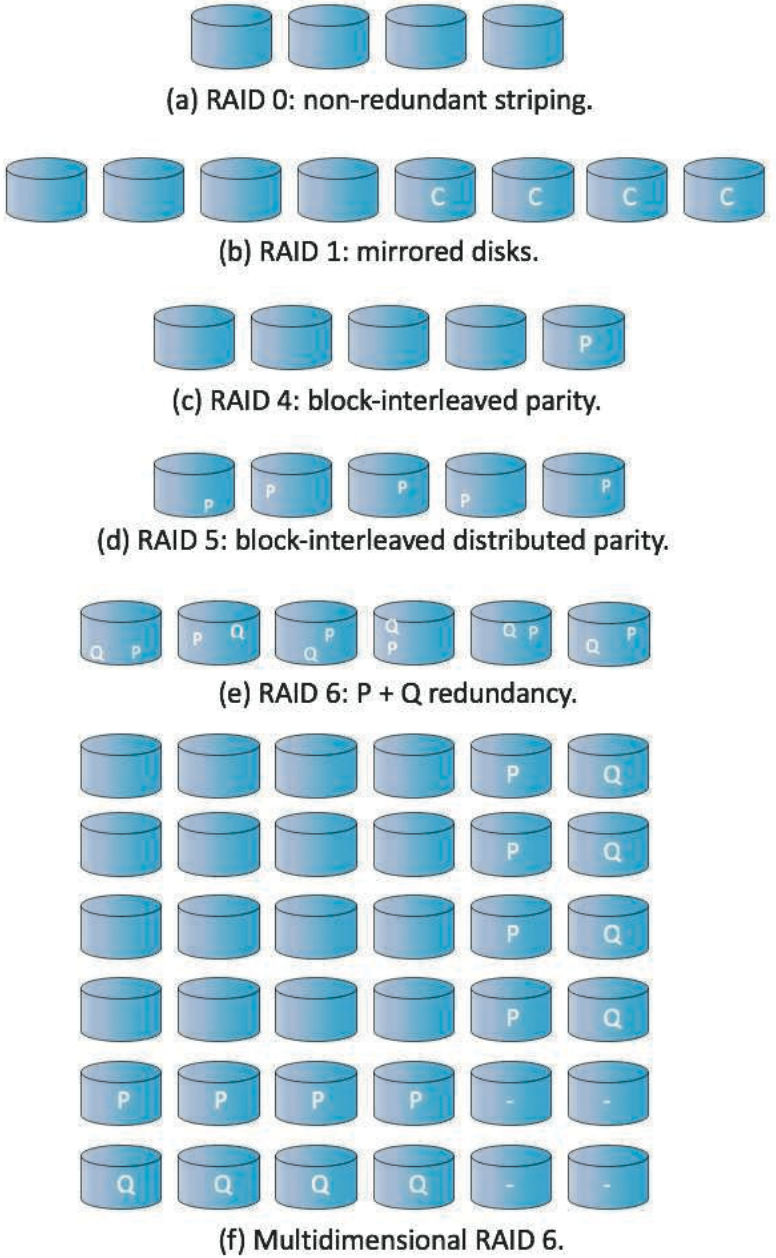


RAID3

- Bit-interleaved parity
  - Data is striped across multiple disks, with one dedicated parity disk that stores the parity information for all the data disks
  - Each write goes to all disks, with each disk storing one bit
  - A parity bit is computed, stored, and used for data recovery
- Example with 4 disks an 1 parity disk
  - Say you store bits 0 1 1 0 on the 4 disks
  - The parity bit stores the XOR of those bits: (((0 xor 1) xor 1) xor 0) = 0
  - Say you lose one bit: 0 ? 1 0
    - You can XOR the remaining bits with the parity bit to recover the lost bit: (((0 xor 0) xor 1) xor 0) = 1
  - Say you lose a different bit: 0 1 1 ?
    - The XOR still works: (((0 xor 1) xor 1) xor 0) = 0
- Bit-level striping increases performance
- XOR overhead for each write (done in hardware)
- Time to recovery is long (a bunch of XOR's)

RAID 4, 5, 6

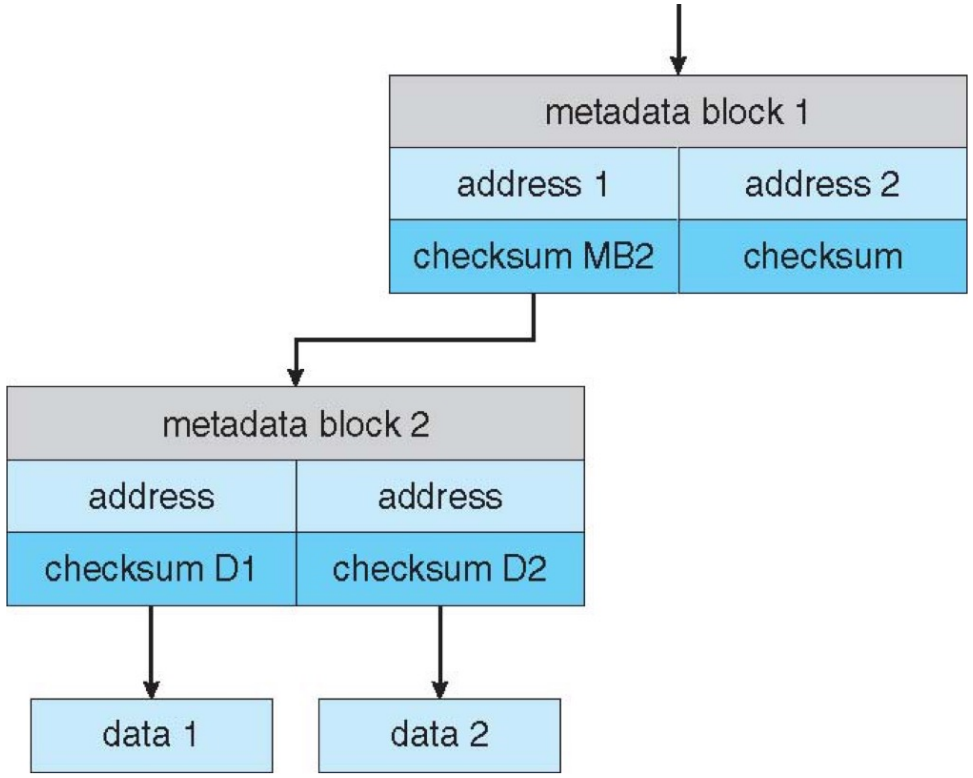
- RAID 4: Basically like RAID 3, but interleaving it with strips (blocks)
  - A (small) read involves only one disk
- RAID 5: Like RAID 4, but parity is spread all over the disks as opposed to having just one parity disk
- RAID 6: extends RAID 5 by adding an additional parity block
  - Has block-level striping with 2 parity blocks



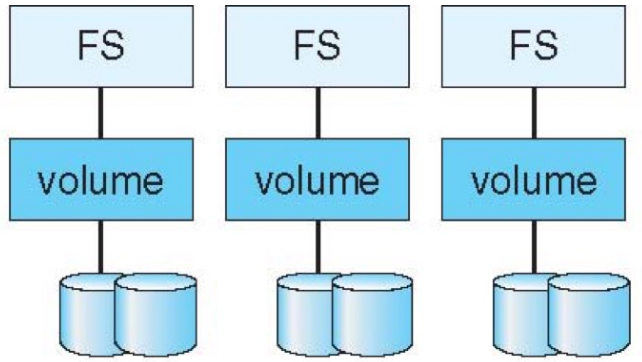
RAID and File Systems

- RAID can only detect/recover from **disk failures**
  - it **does not** prevent or detect **data corruption** or other errors
- File systems like Solaris ZFS add additional checks to detect errors
  - ZFS adds checksums to all **FS data and metadata**
    - checksum is collocated with pointer to the data/metadata
    - can detect and correct data and metadata corruption
  - ZFS allocates disks in pools, instead of volumes or partitions
    - file systems within a pool share that pool, allocate/free space from/to pool

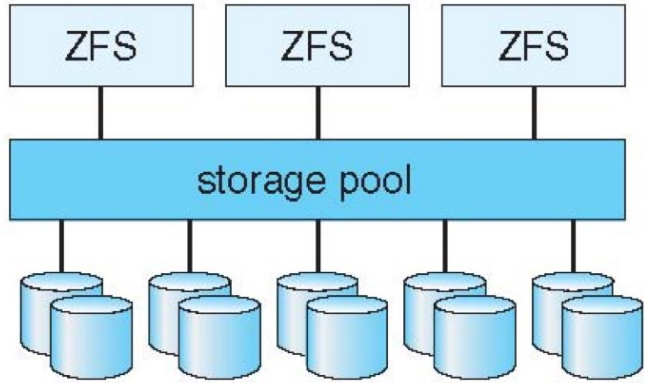
ZFS Checksums



Traditional and Pooled Storage



(a) Traditional volumes and file systems.



(b) ZFS and pooled storage.

Takeaway

- Disk structure
- Disk scheduling
  - FCFS, SSTF, SCAN, C-SCAN, LOOK, C-LOOK
- RAID
  - 0-6