Mass-Storage Systems

Politecnico di Torino

Department of Control and Computer Engineering

System Programming - Sarah Azimi

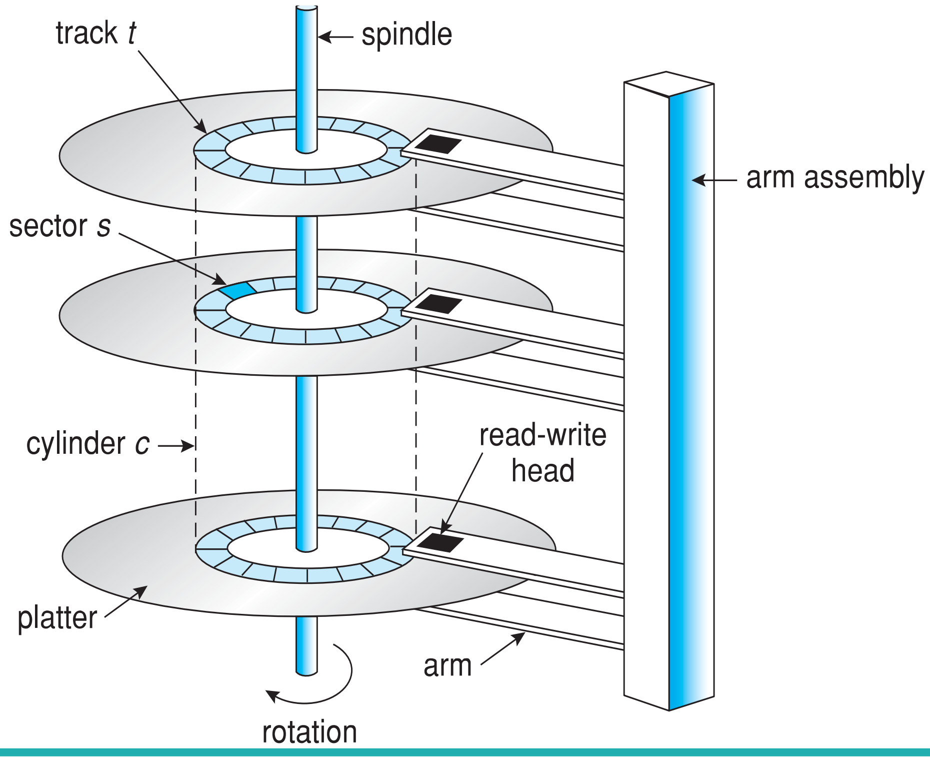
CAD & Reliability Group
DAUIN– Politecnico di Torino

Overview of Mass Storage Structure

- The main mass-storage system in modern computers is secondary storage, which is usually provided by hard disk drivers and nonvolatile memory devices.
- The secondary storage for modern computers is **hard disk drives** (HDDs) and **nonvolatile memory** (NVM) devices.
- HDDs spin platters of magnetically-coated material under moving read-write heads.
  - Drives rotate at 60 to 250 times per second
  - Transfer rate is the rate at which data flow between the drive and the computer
  - Positioning time (random-access time) is the time to move the disk arm to the desired cylinder (seek time) and the time for the desired sector to rotate under the disk head (rotational latency)
  - Head crash results from disk head making contact with the disk surface -- That's bad
- Disks can be removable

Hard Disk Drives

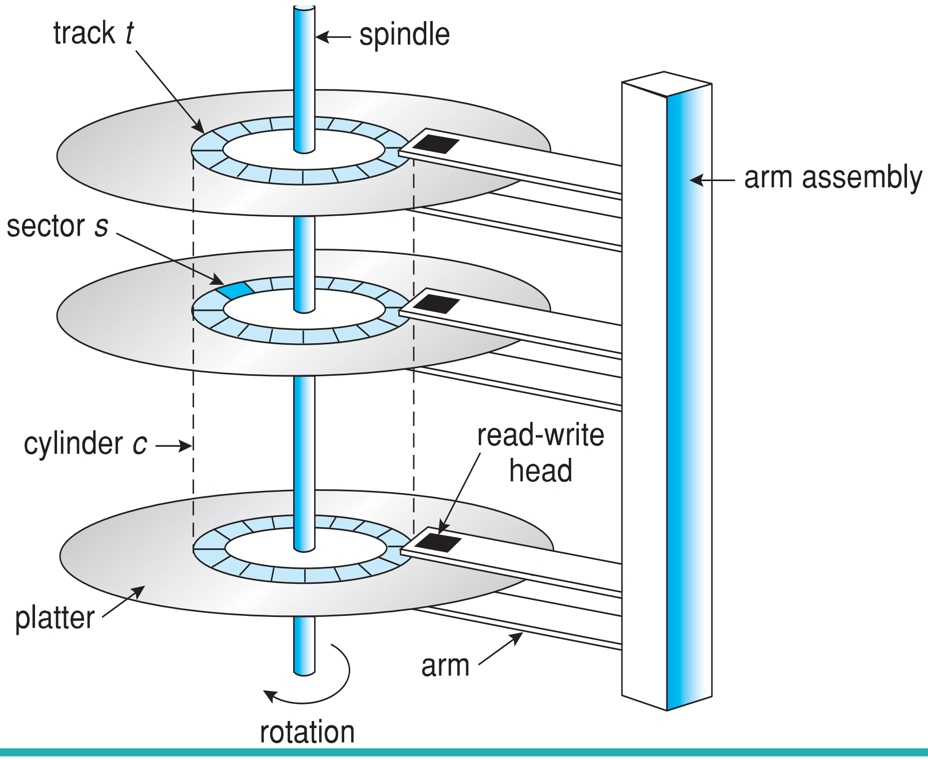
- HDDs are disk platters, with the two surfaces covered with a magnetic material.
- Storing information by recording it magnetically on the platters and reading information by detecting the magnetic pattern on the platters.
  - The surface of the platter is divided into circular **tracks**, which are divided into **sectors**.
  - The set of tracks at a given arm position is called a **cylinder**.



Hard Disk Drives

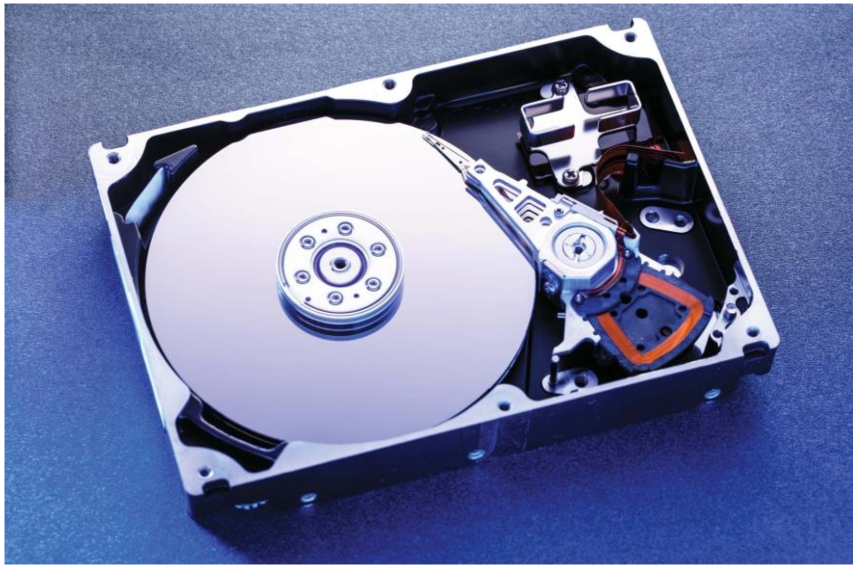
- HDDs are disk platters, with the two surfaces covered with a magnetic material.
- Storing information by recording it magnetically on the platters, and reading information by detecting the magnetic pattern on the platters.

- The **transfer rate** is the rate at which data flow between the driver and the computer.
- The **positioning time** or the **random-access time** consists of two times:
- The **seek time**, the necessity to move the disk arm to the desired cylinder
- The **rotational latency**, the time necessary for the desired sector to rotate to the disk head.
- Drivers rotate at 60 to 250 times per second.



Hard Disks

- Platters range from .85" to 14" (historically)
  - Commonly 3.5", 2.5", and 1.8"
- Range from 30GB to 3TB per drive
- Performance
  - **Transfer Rate – theoretical – 6 Gb/sec**
  - Effective Transfer Rate – real – 1Gb/sec
  - **Seek time from 3ms to 12ms** – 9ms common for desktop drives
  - Average seek time measured or calculated based on 1/3 of tracks
  - Latency based on spindle speed
    - 1 / (RPM / 60) = 60 / RPM
  - Average latency = ½ latency



Hard Disk Performance

- **Access Latency = 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 + (amount to transfer / transfer rate) + controller overhead
- For example, to transfer a **4KB block** on a **7200 RPM** disk with a **5ms** average **seek time**, 1Gb/sec transfer rate with a .1ms controller overhead =
  - 5ms + 4.17ms + 0.1ms + transfer time
  - Transfer time = 4KB / 1Gb/s * 8Gb / GB * 1GB / 1024^2KB = 32 / (1024^2) = 0.031 ms
  - Average I/O time for 4KB block = 9.27ms + .031ms = 9.301ms

  - 4KB = 4*8*1024 = 32768bit = 0.032768Gb
  - Transfer time = (1Gb/sec) / 0.032768 = 0.031 ms

  - Time per rotation = 60 seconds / 7200 RP = 8.333 ms

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

Nonvolatile Memory Devices

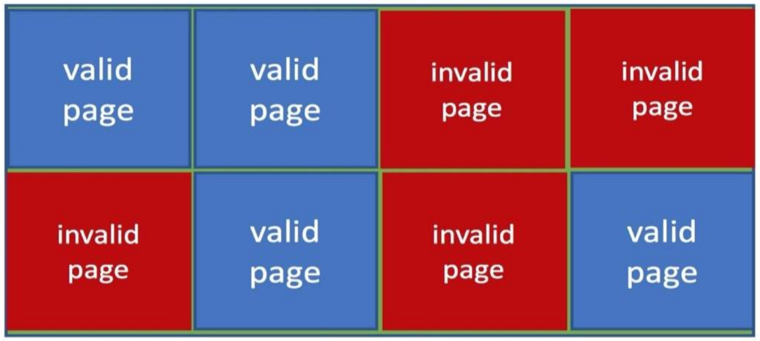
- Nonvolatile Memory (NVM) devices are electrical rather than mechanical.
- NVM is used in a disk-drive-like container called a **solid-state disk** (SSD)
- Other forms include USB drives (thumb drive, flash drive), DRAM disk replacements, surface-mounted on motherboards, and main storage in devices like smartphones
  - NVM can be **more reliable** than HDDs because they have no moving part
  - They can be faster because they have no seek time or rotational latency.
  - They consume less power.
  - More expensive per MB
  - Maybe have a shorter life span – need careful management
  - Less capacity
  - Over time, NVM devices' capacity has increased faster than HDD capacity and their price dropped so their use is increasing dramatically.

Nonvolatile Memory Devices

- Have characteristics that present challenges
- They can be Read and written in “page” increments (think sector) but can’t be overwritten in place
  - The NAND cells have to be erased first
  - The erasure which occurs in a block increment that is several pages in size, take much more time than a read or write.
  - Can only be erased a limited number of times before worn out – ~ 100,000
  - Life span measured in **Drive Writes Per Day (DWPD)** that measures how many times the driver capacity can be written per day before it fails.
    - A 1TB NAND drive with a rating of 5DWPD is expected to have 5TB per day written within the warrantee period without failing

NAND Flash Controller Algorithms

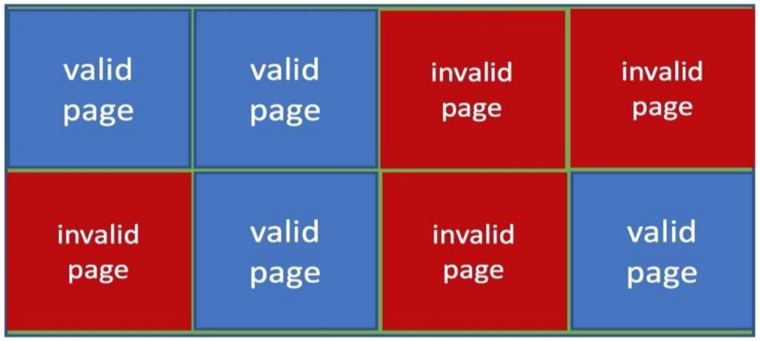
- Since NAND semiconductors cannot be overwritten once written, they are usually pages containing invalid data.
- With no overwrite, pages end up with a mix of valid and invalid data.
- To track which logical blocks are valid, the controller maintains a **flash translation layer (FTL)** table
- This table maps which physical pages contain currently valid logical blocks.
- Also implements **garbage collection** to free invalid page space (valid data could be copied to other locations, freeing up blocks that could be erased and receive the write.



NAND block with valid and invalid pages

NAND Flash Controller Algorithms

- Where would the garbage collector store the good data? NMV devices use **over-provisioning** >> setting aside a number of pages as an area always available to write to.
- Each cell has a lifespan, so **wear leveling is** needed to write equally to all cells (if some blocks are erased repeatedly while others are not, the frequently erased blocks will wear out faster than the others, and the entire device will have a shorted lifespan,)



NAND block with valid and invalid pages

Volatile Memory

- DRAM frequently used as a mass-storage device
  - Not technically secondary storage because volatile, but can have file systems, and be used like very fast secondary storage
- **RAM drives** (with many names, including RAM disks) present as raw block devices, commonly file system formatted
- Computers have buffering, caching via RAM, so why RAM drives?
  - Caches / buffers allocated / managed by programmer, operating system, hardware
  - RAM drives under user control
  - Found in all major operating systems
    - Linux /dev/ram, macOS diskutil to create them, Linux /tmp of file system type tmpfs
- Used as high-speed temporary storage
  - Programs could share bulk date, quickly, by reading/writing to RAM drive

Magnetic Tape

- Was early secondary-storage medium
  - Evolved from open spools to cartridges
- Relatively permanent and holds large quantities of data
- Access time slow
- Random access ~1000 times slower than disk
- Mainly used for backup, storage of infrequently-used data, transfer medium between systems
- Kept in spool and wound or rewound past read-write head
- Once data under head, transfer rates comparable to disk
  - 140MB/sec and greater
- 200GB to 1.5TB typical storage
- Common technologies are LTO-{3,4,5} and T10000

Disk Attachment

- A secondary storage device is attached to a computer by the system bus or an I/O bus.
- Several kind of buses are available including **Advanced Technology Attachment (ATA), Serial ATA (SATA), eSATA, Serial Attached SCSI (SAS), Universal Serial Bus (USB), and Fiber Channel (FC)**.
- The most common one is SATA
- Since NVM devices are faster than HDDs, the industry created a special, fast interface for NVM devices called **NVM express (NVMe)**, connecting directly the device to the system PCI bus.
- Data transfers on a bus carried out by special electronic processors called **controllers** (or **host-bus adapters, HBAs**).
  - Host controller on the computer end of the bus, device controller on device end
  - Computer places command on host controller, using memory-mapped I/O ports
    - Host controller sends messages to device controller
    - Data transferred via DMA between device and computer DRAM

Address Mapping

- Storage devices are addressed as large 1-dimensional arrays of **logical blocks**, where the logical block is the smallest unit of transfer
  - Each logical blocks maps to a physical sector or semiconductor page.
- The 1-dimensional array of logical blocks is mapped into the sectors of the disk sequentially
  - Sector 0 is the first sector of the first track on the outermost cylinder
  - Mapping proceeds in order through that track, then the rest of the tracks in that cylinder, and then through the rest of the cylinders from outermost to innermost
  - Logical to physical address should be easy
    - Except most drivers have some defective sectors , but the mapping hides this by substituting spare sectors from elsewhere on the drive. The logical block address stays sequential, but the physical sector location changes.
    - The number of sectors per track is not a constant on some drives.

HDD Scheduling

- The operating system is responsible for using hardware efficiently — for the disk drives, this means **minimizing access time** and **maximizing data transfer bandwidth.**
- Access time has two major components: Seek time and rotational latency
  - Minimize seek time
  - **Seek time ≈ seek distance**
- Disk **bandwidth** is the total number of bytes transferred, divided by the total time between the first request for service and the completion of the last transfer.
- We can improve both the access time and the bandwidth by managing the order in which storage I/O requests are serviced.

Disk Scheduling

- There are many sources of disk I/O request
  - OS
  - System processes
  - User processes
- When a process needs I/O to or from the drive, it issues a system call to the operating system which request for information such as:
  - Whether this operation is input or output
  - What memory address for the transfer is
  - The amount of data to transfer
- If the desired drive and controller are available, the request can be services immediately.
- If they are busy, any new requests for service will be placed in the queue of pending request.
  - Optimization algorithms only make sense when a queue exists

Disk Scheduling (Cont.)

- Note that driver controllers have small buffers and can manage a queue of I/O requests (of varying "depth")
- Several algorithms exist to schedule the servicing of disk I/O requests
- The analysis is true for one or many platters
- We illustrate scheduling algorithms with a request queue (0-199)

98, 183, 37, 122, 14, 124, 65, 67

Head pointer 53

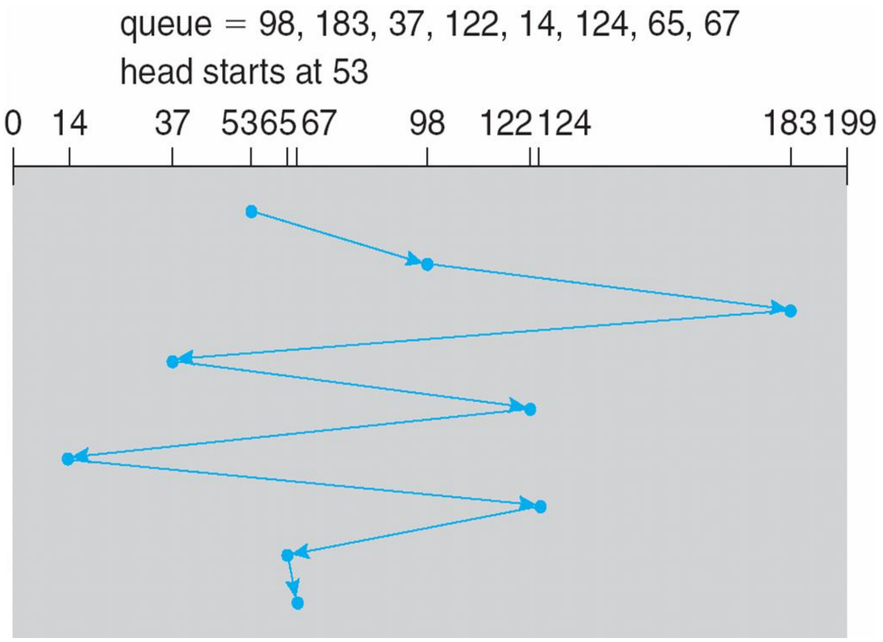
FCFS

- The simplest for of disk scheduling is First Come First Served which is fair but not always fastest.
- We illustrate scheduling algorithms with a request queue (0-199)

98, 183, 37, 122, 14, 124, 65, 67

Head pointer 53

Illustration shows total head movement of 640 cylinders



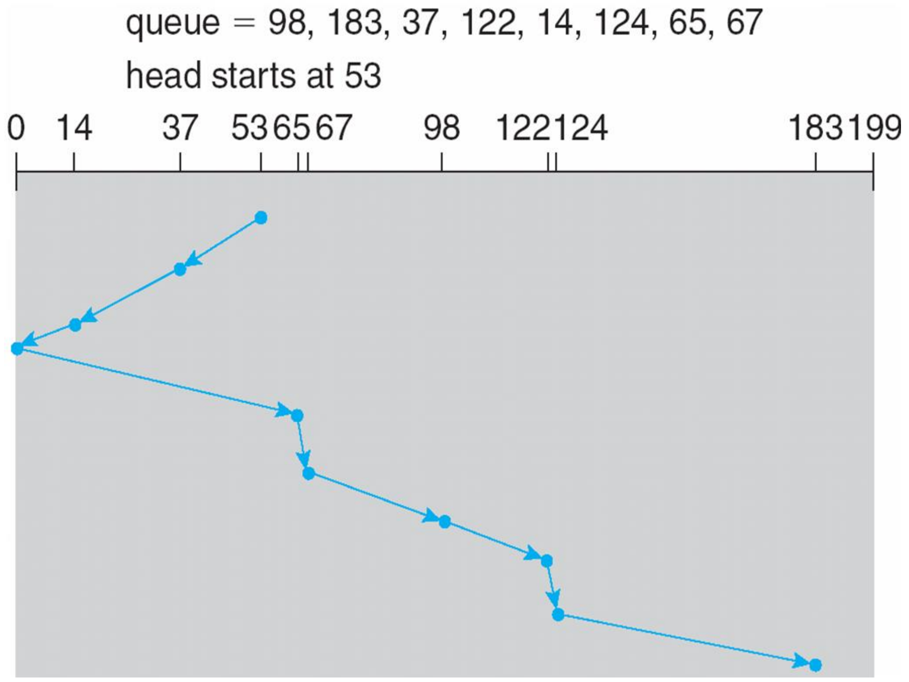
- The total head movement of 640 cylinders.
- The swing from 122 to 14 and then back to 124 illustrated the problem.
- If the requests for cylinder 37 and 14 could be serviced together, before or after the requests for 122 and 124, the total head movement could be decreased, improving the performance.

SCAN

- The disk arm starts at one end of the disk, and moves toward the other end, servicing requests until it gets to the other end of the disk, where the head movement is reversed, and servicing continues.
- **SCAN algorithm** sometimes called the **elevator algorithm**
- Illustration shows total head movement of 236 cylinders
- But note that if requests are uniformly dense, largest density at another end of disk and those wait the longest

SCAN

- We need to know the direction of head movement in addition to the head's current position.
- Assuming the disc arm is moving toward o and that the initial head position is again 53, the head will serve 37, 14, 65, 67, 98, 122, 124 and 183.

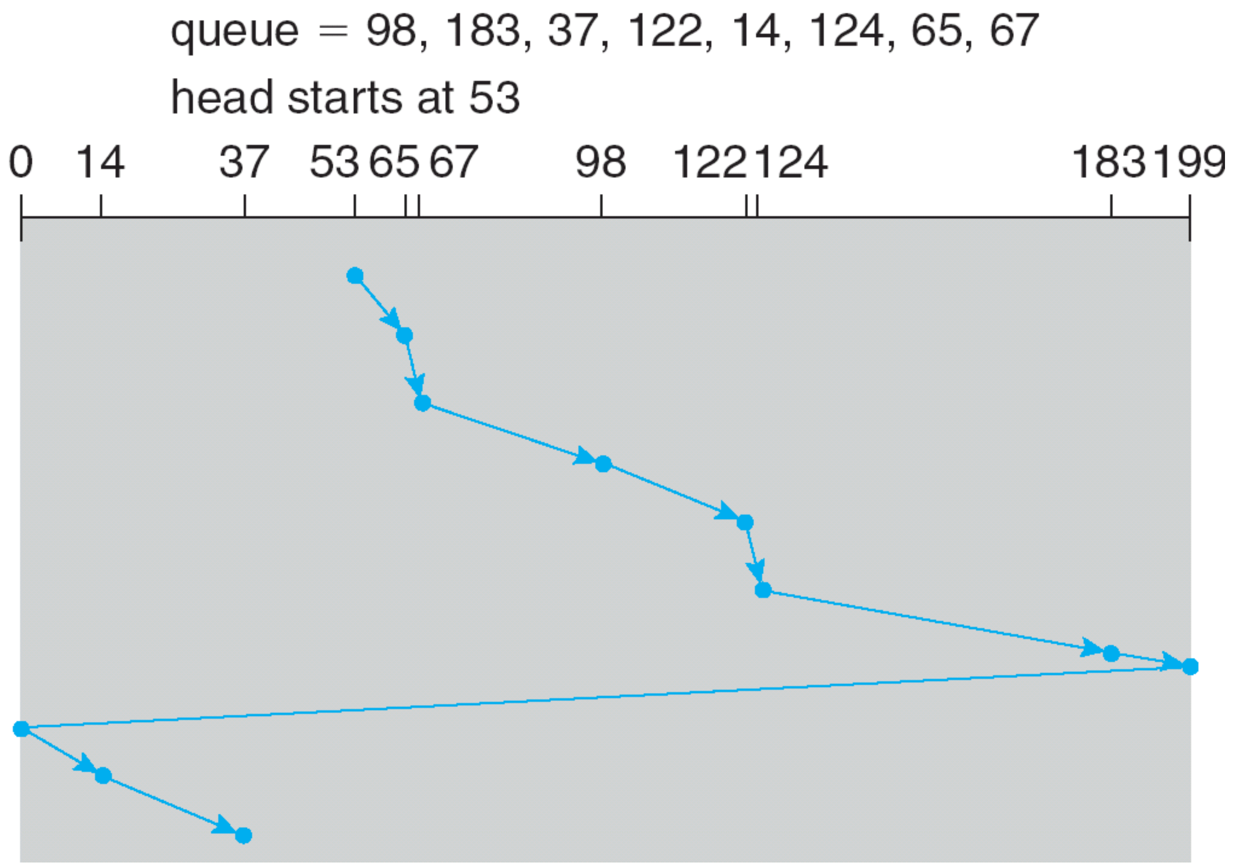


C-SCAN

- Provides a more uniform wait time than SCAN
- The head moves from one end of the disk to the other, servicing requests as it goes
- When it reaches the other end, however, it immediately returns to the beginning of the disk, without servicing any requests on the return trip
- Treats the cylinders as a circular list that wraps around from the last cylinder to the first one

C-SCAN (Cont.)

- We need to know the direction of head movement in addition to the head's current position.
- Assuming the disc arm is moving 0 to 199 and that the initial head position is again 53, the head will serve as:



Selecting a Disk-Scheduling Algorithm

- FCFS is common and has a natural appeal
- SCAN and C-SCAN perform better for systems that place a heavy load on the disk
  - Since they cause Less starvation, but still possible
- To avoid starvation Linux implements **deadline scheduler**
  - Maintains separate read and write queues, gives read priority
    - Because processes more likely to block on read than write
  - Deadline Implements four queues: 2 x read and 2 x write
    - 1 read and 1 write queue sorted in LBA order, essentially implementing C-SCAN
    - 1 read and 1 write queue sorted in FCFS order
    - All I/O requests sent in batch sorted in that queue's order
    - After each batch, checks if any requests in FCFS older than configured age (default 500ms)
      - If so, LBA queue containing that request is selected for next batch of I/O
- The deadline I/O scheduler is the default in the Linux RedHat 7 distribution, but RHEL 7 also includes two others: **NOOP** and **completely fair queueing** scheduler **(CFQ)** also available, defaults vary by storage device

NVM Scheduling

- No disk heads or rotational latency but still room for optimization
- In RHEL 7 **NOOP** (no scheduling) is used but adjacent LBA requests are combined
  - NVM best at random I/O, HDD at sequential
  - Throughput can be similar
  - **Input/Output operations per second (IOPS)** much higher with NVM (hundreds of thousands vs hundreds)
  - But **write amplification** (one write, causing garbage collection and many read/writes) can decrease the performance advantage

Error Detection and Correction

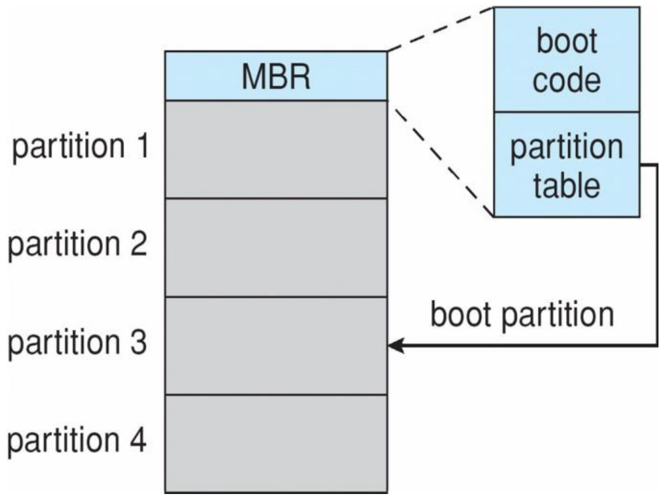
- Fundamental aspect of many parts of computing (memory, networking, storage)
- **Error detection** determines if a problem has occurred (for example a bit flipping)
  - For example, a bit in DRAM changed from a 1 to a 1.
  - If detected, can halt the operation before it is propagated
  - Detection frequently done via parity bit that records whether the number of bits in the byte set to 1 is even or odd.
- Parity is one form of **checksum** – uses modular arithmetic to compute, store, compare values of fixed-length words
  - Another error-detection method common in networking is **Cyclic Redundancy Check (CRC)** Which uses hash function to detect multiple-bit errors
- **Error-correction code (ECC)** not only detects, but can correct some errors
  - Soft errors correctable, hard errors detected but not corrected

Storage Device Management

- A new storage device is a blank slate; it is a set of uninitialized semiconductor storage cells.
- Before a storage device can store data, it must be divided into sectors that the controller can read and write.
- This process is called **Low-level formatting, or physical formatting** — Dividing a disk into sectors that the disk controller can read and write
    - Each sector can hold header information, data area, trailor, plus error correction code (**ECC**)
    - Usually 512 bytes of data but can be selectable
- To use a disk to hold files, the operating system still needs to record its own data structures on the disk
    - **Partition** the disk into one or more groups of blocks or pages. The partition information is written in a fixed format at a fixed location on the storage device.
    - **Logical formatting** or “making a file system” in which the operating system stores the initial file-system data structures onto the device. The data structure may include maps of free and allocated space and an initial empty directory.
    - To increase efficiency most file systems group blocks into **clusters**
        - Disk I/O done in blocks
        - File I/O done in clusters

Storage Device Management (cont.)

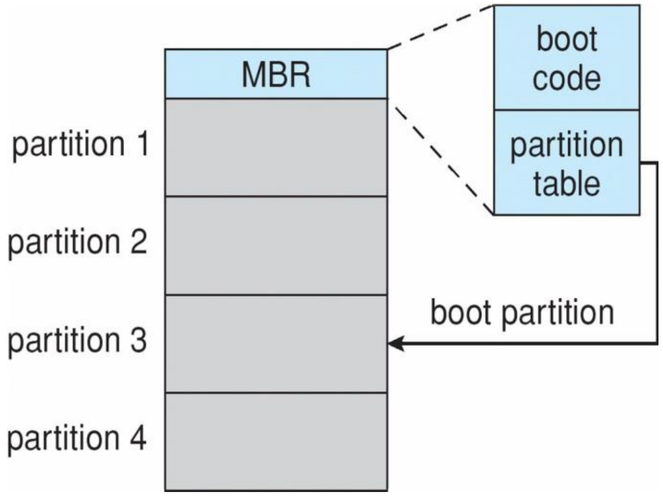
- For a computer to start running, it must have an initial program to run.
- This initial **bootstrap loader** is stored in NVM flash memory firmware on the system motherboard and mapped to known memory location.
- It initializes all aspects of the system, from CPU registers to device controllers and the content of the main memory.
- The full bootstrap program is stored in **boot blocks** at the fixed location.
- A device that has a boot partition is called a **boot disk** or **system disk**.



Booting from secondary storage in Windows

Storage Device Management (cont.)

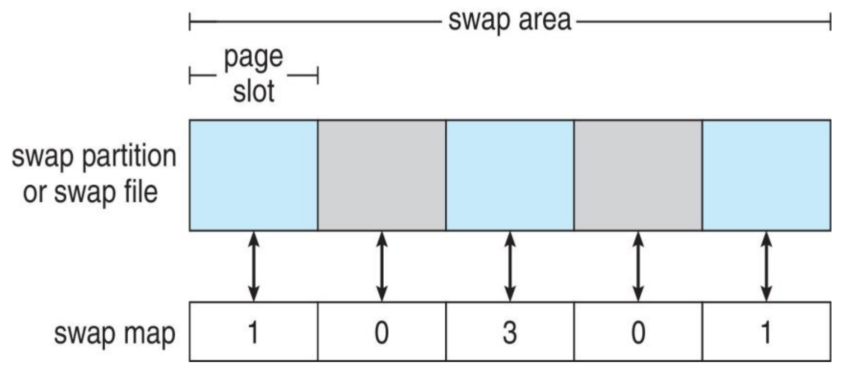
- An example of boot process in windows:
- Drive is divided into partition which is called **boot partition** contains the operating system and device drivers.
- The windows system places its boot code in the first logical block on the hard disk or first page of the NVM device, **master boot record** or **MBR**.
- Booting begins by running code that is resident in the systems firmware



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 is not large enough for all processes
- Operating system provides swap space management to provide the best throughput for the virtual memory system.
  - Secondary storage slower than DRAM, so important to optimize performance
  - Usually multiple swap spaces possible – decreasing I/O load on any given device
  - Best to have dedicated devices
  - Can be in raw partition or a file within a file system (for convenience of adding)
  - Data structures for swapping on Linux systems:

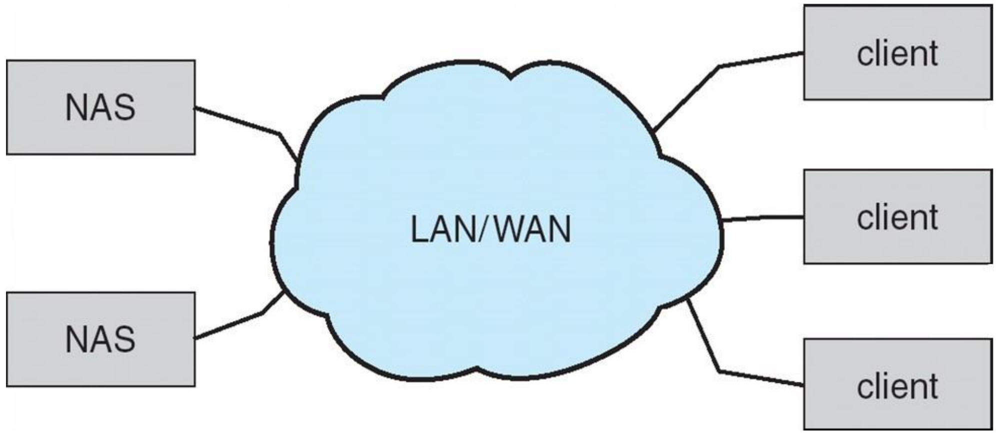


Storage Attachment

- Computers access storage in three ways
  - host-attached
  - network-attached
  - cloud
- Host attached access through local I/O ports, using one of several technologies
  - To attach many devices, use storage busses such as USB, firewire, thunderbolt
  - High-end systems use fibre channel (FC)
    - High-speed serial architecture using fibre or copper cables
    - Multiple hosts and storage devices can connect to the FC fabric

Network-Attached Storage

- Network-attached storage (NAS) is storage made available over a
- network rather than over a local connection (such as a bus)
    - Remotely attaching to file systems
- NFS and CIFS are common protocols
- Implemented via remote procedure calls (RPCs) between
- host and storage over typically TCP or UDP on IP network
- **iSCSI** protocol uses IP network to carry the SCSI protocol
    - Remotely attaching to devices (blocks)



Cloud Storage

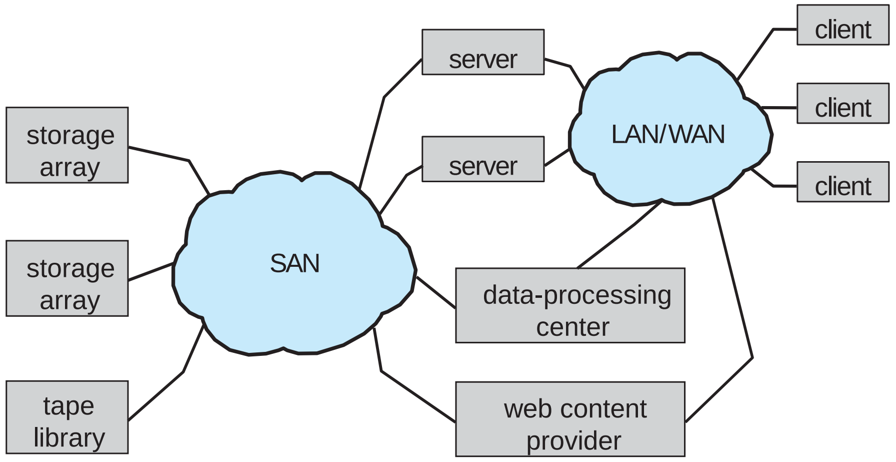
- Similar to NAS, provides access to storage across a network
  - Unlike NAS, accessed over the Internet or a WAN to remote data center
- NAS presented as just another file system, while cloud storage is API based, with programs using the APIs to provide access
  - Examples include Dropbox, Amazon S3, Microsoft OneDrive, Apple iCloud
  - Use APIs because of latency and failure scenarios (NAS protocols wouldn’t work well)

Storage Array

- Can just attach disks, or arrays of disks
- Avoids the NAS drawback of using network bandwidth
- Storage Array has controller(s), provides features to attached host(s)
  - Ports to connect hosts to array
  - Memory, controlling software (sometimes NVRAM, etc)
  - A few to thousands of disks
  - RAID, hot spares, hot swap (discussed later)
  - Shared storage -> more efficiency
  - Features found in some file systems
    - Snaphots, clones, thin provisioning, replication, deduplication, etc

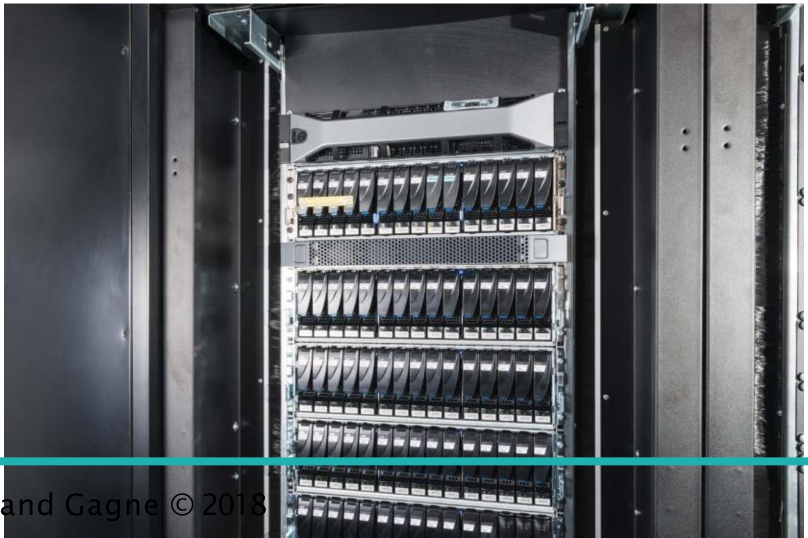
Storage Area Network

- Common in large storage environments
- Multiple hosts attached to multiple storage arrays – flexible



Storage Area Network (Cont.)

- SAN is one or more storage arrays
  - Connected to one or more Fibre Channel switches or **InfiniBand (IB)** network
- Hosts also attach to the switches
- Storage made available via **LUN Masking** from specific arrays to specific servers
- Easy to add or remove storage, add new host and allocate it storage
- Why have separate storage networks and communications networks?
- Consider iSCSI, FCOE


A Storage Array

RAID Structure

- Storage devices have continued to get smaller and cheaper, so it is economic to attach many drivers to computer systems.
- Having a large number of drivers in a system presents opportunities for improving the rate at which data can be read or written as well as improving the reliability of data storage
  - Redundant information can be stored on multiple drives.
- A variety of disk organization techniques called **redundant arrays of independent disks (RAIDs).**

RAID Structure

- **RAID – redundant array of inexpensive independent disks**
  - multiple disk drives provides reliability via **redundancy**
- Increases the **Mean Time To Failure (MTTF)**
  - Suppose that Mean Time To Failure (MTTF) of a single disk is 100,000 hours, then the MTTF of some disk in an array of 100 disk will be 100,000/100 = 1,000 hours or 41.66 days.
  - Storing one copy of data >> each disk failure will result in loss of a significant amount of data.
- The solution to the problem of reliability is to introduce **Redundancy**.
  - Storing the disk failure to rebuild the lost information that is not normally needed but can be used in the event of disk failure to rebuild the lost information.
- The simplest approach to introduce redundancy is duplication of every drive – **Mirroring**.
  - A logical disk consists of two physical drives, and every write is carried out on both drives – **mirrored volume**.

RAID Structure

- The MTBF of a mirrored volume depends on two factors:
  - MTBF of individual drives
  - **Mean Time To Repair (MTTR):** the time it takes to replace a failed drive and restore the data on it.

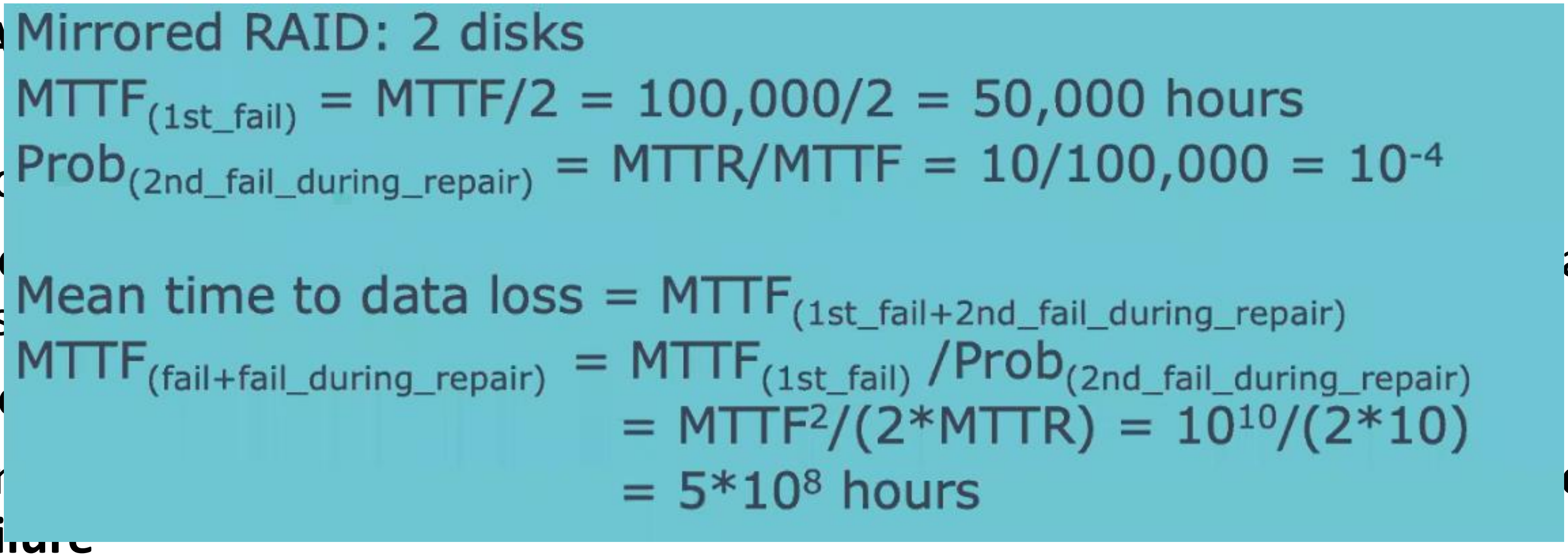
  - Suppose that the failures of the two drives are independent, if the MTBF of a single drive is 100,000 hours, and the MTTR is 10 hours, the Mean Time to Data Loss of a mirrored drive system is $100{,}000^2/(2*10) = 500*10^6$ hours or 57,000 years.

RAID Structure

- **RAID – redundant array of inexpensive independent disks**
  - multiple disk drives provides reliability via **redundancy**
- Increases the **mean time to failure (MTTF)**
- **Mean time to repair** – exposure time when another failure could cause data loss
- **Mean time to data loss** based on above factors
- If mirrored disks fail independently, consider disk with 100,000 **mean time to failure**
- and 10 hour mean time to repair
  - Mean time to data loss is 100, 0002 / (2 ∗ 10) = 500 ∗ 106 hours, or 57,000 years!
- Frequently combined with **NVRAM** to improve write performance
- Several improvements in disk-use techniques involve the use of multiple disks working cooperatively

RAID Structure

- **RA**
  -
- Inc
- **M** ata los
- **M**
- If r **e to fai**
- and 10 hour mean time to repair
  - Mean time to data loss is 100, 0002 / (2 ∗ 10) = 500 ∗ 106 hours, or 57,000 years!
- Frequently combined with **NVRAM** to improve write performance
- Several improvements in disk-use techniques involve the use of multiple disks working cooperatively



Mirrored RAID: 2 disks

$MTTF_{(1st\_fail)} = MTTF/2 = 100{,}000/2 = 50{,}000$ hours

$Prob_{(2nd\_fail\_during\_repair)} = MTTR/MTTF = 10/100{,}000 = 10^{-4}$

Mean time to data loss = $MTTF_{(1st\_fail+2nd\_fail\_during\_repair)}$

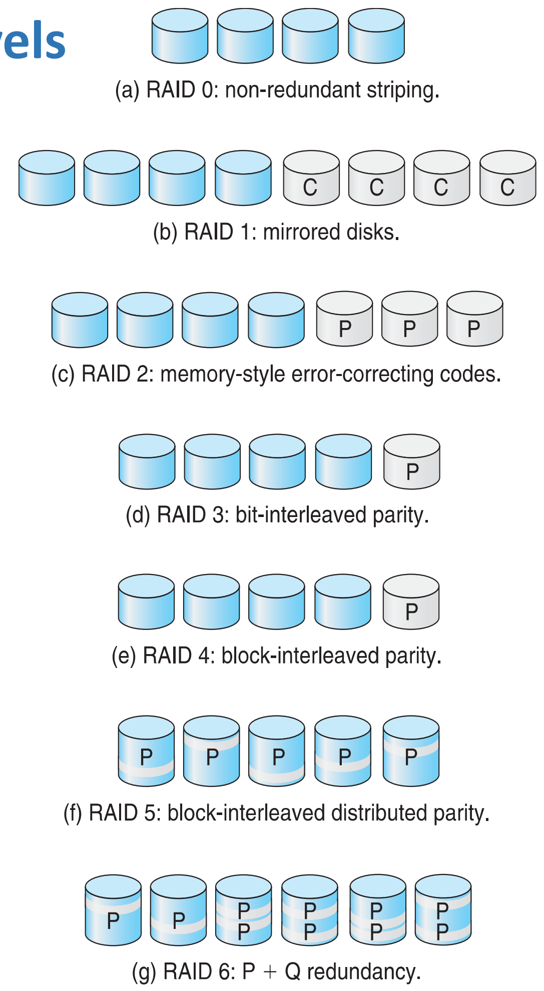
$$MTTF_{(fail+fail\_during\_repair)} = MTTF_{(1st\_fail)} / Prob_{(2nd\_fail\_during\_repair)}$$
$$= MTTF^2/(2*MTTR) = 10^{10}/(2*10)$$
$$= 5*10^8 \text{ hours}$$

RAID for improvement in performance

- With mirroring, the rate at which read requests can be handled is doubles, since read requests can be sent to either drive.
  - The transfer rate of each read is the same as in a single-drive system but the number of reads per unit time has doubles.
- Improving the transfer rate by **stripping data** across the drives.
  - **Bit-level stripping**: splitting the bits of each byte across multiple drives.
    - If we have an array of eight drives, we write but I of each byte to drive i.
  - **Block-level stripping**: blocks of files are stripped across multiple drives.
    - With n drives, block I of a file goes to drive (i mod n) + 1.
- Benefits of parallelism in a storage system:
  - Increasing the throughput of multiple small accesses
  - Reducing the response time of large accesses.

RAID Levels

- Mirroring provides high reliability, but it is expensive.
- Stripping provides high data-transfer rate, but it doesn't improve reliability.
- Data stripping combined with redundancy : **RAID Levels**
  - **RAID Level 0**
  - **RAID Level 1**
  - **RAID Level 4**
  - **RAID Level 5**
  - **RAID Level 6**
  - **Multidimensional RAID Level 6**
  - **RAID Levels 0 + 1 and 1 + 0**



(a) RAID 0: non-redundant striping.

(b) RAID 1: mirrored disks.

(c) RAID 2: memory-style error-correcting codes.

(d) RAID 3: bit-interleaved parity.

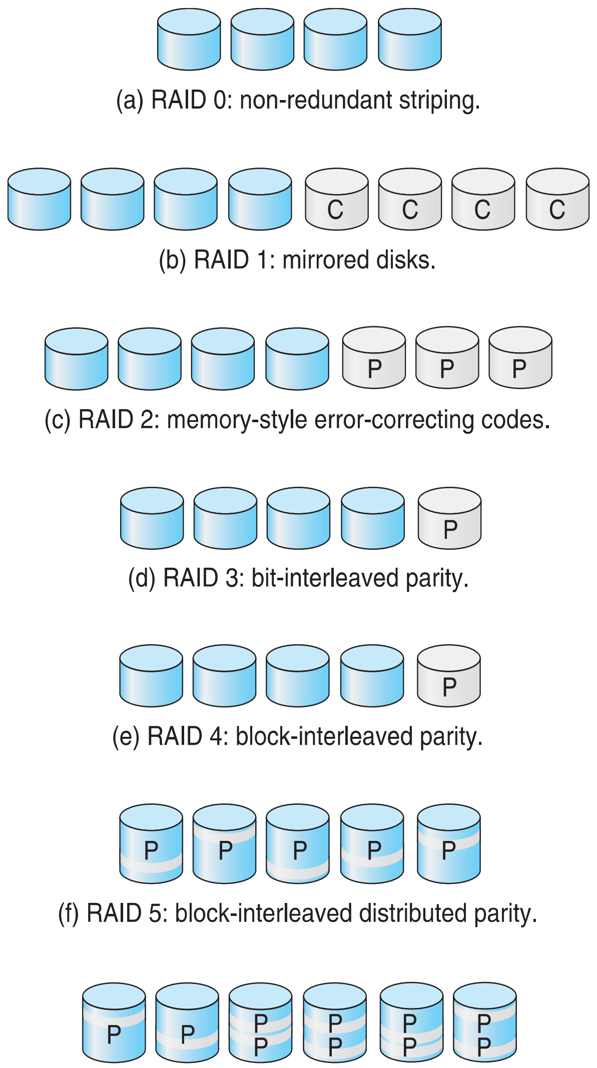
(e) RAID 4: block-interleaved parity.

(f) RAID 5: block-interleaved distributed parity.

(g) RAID 6: P + Q redundancy.

RAID Levels

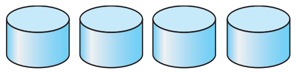
- RAID Level 0
  - Refers to drive arrays with stripping at the level of blocks but without any redundancy
- RAID Level 1
  - Refers to drive monitoring (mirrored organization)
- RAID Level 4
  - Known as memory-style error-correcting code (ECC) organization.
  - ECC can be directly used in storage arrays via stripping of blocks across drives.
    - The first data block of a sequence of writes can be stored in drive 1, the second block in drive 2 and so on. The Nth block is stored in drive N, the error-correction calculation results of these blocks is stored on drive N+1.

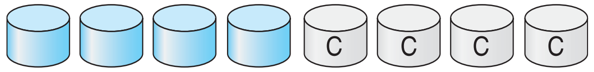


(a) RAID 0: non-redundant striping.
(b) RAID 1: mirrored disks.
(c) RAID 2: memory-style error-correcting codes.
(d) RAID 3: bit-interleaved parity.
(e) RAID 4: block-interleaved parity.
(f) RAID 5: block-interleaved distributed parity.
(g) RAID 6: P + Q redundancy.

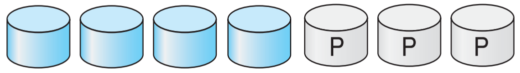
RAID Levels

- RAID Level 5
  - Known as block-interleaved distributed parity, spreading data and parity among all N+1 drives, rather than storing data in N drives and parity in one drive.
- RAID Level 6
  - Called p*Q redundancy scheme is like RAID level 5 but stores extra redundant information to guard against multiple drive failures.
- Multidimensional RAID Level 6
  - Logically arranging drives into rows and columns and implements RAID level 6 both horizontally along the rows and vertically down the columns.
- RAID Levels 0 + 1 and 1 + 0
  - Combination of RAID levels 0 and 1 for when both reliability and performance are important.

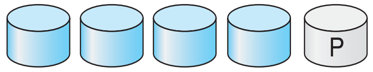

(a) RAID 0: non-redundant striping.



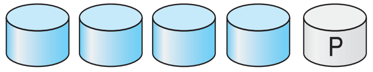
(b) RAID 1: mirrored disks.



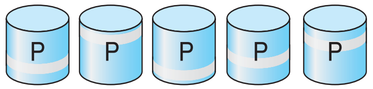
(c) RAID 2: memory-style error-correcting codes.



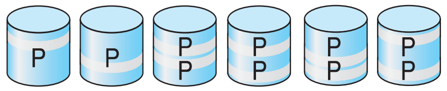
(d) RAID 3: bit-interleaved parity.



(e) RAID 4: block-interleaved parity.

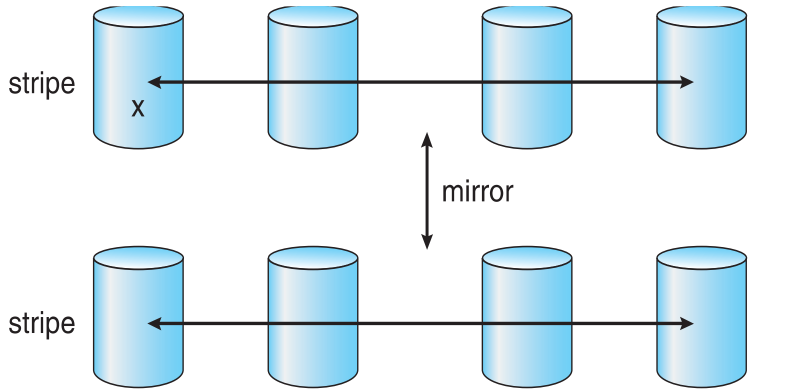


(f) RAID 5: block-interleaved distributed parity.

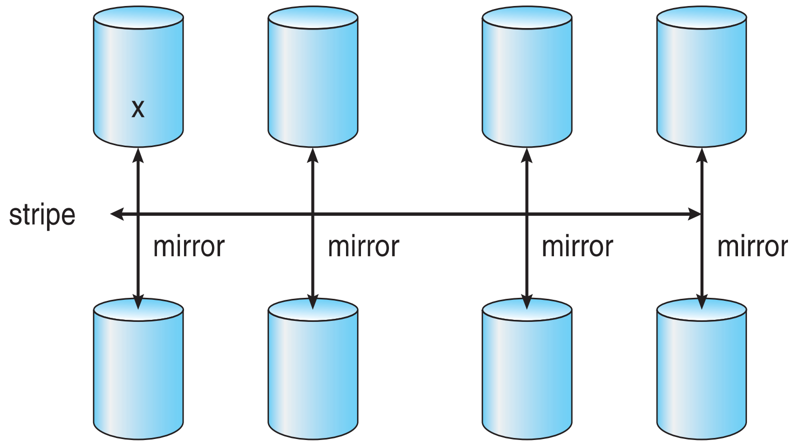


(g) RAID 6: P + Q redundancy.

RAID (0 + 1) and (1 + 0)



a) RAID 0 + 1 with a single disk failure.



b) RAID 1 + 0 with a single disk failure.

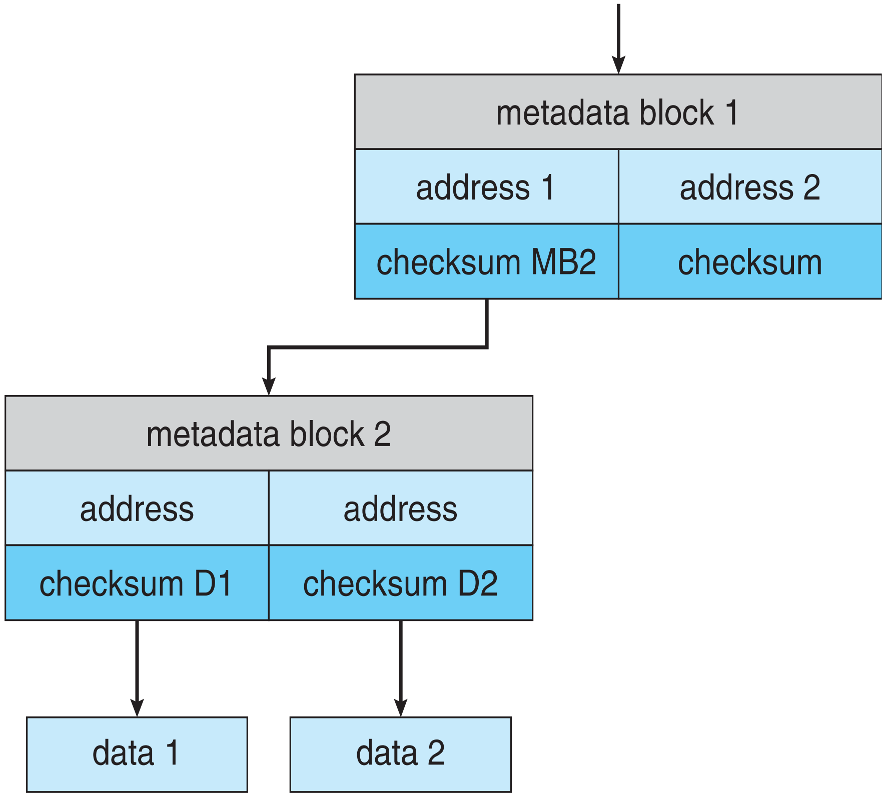
Other Features

- Regardless of where RAID implemented, other useful features can be added
- **Snapshot** is a view of file system before a set of changes take place (i.e. at a point in time)
  - More in Ch 12
- **Replication** is automatic duplication of writes between separate sites
  - For redundancy and disaster recovery
  - Can be synchronous or asynchronous
- Hot spare disk is unused, automatically used by RAID production if a disk fails to replace the failed disk and rebuild the RAID set if possible
  - Decreases mean time to repair

Extensions

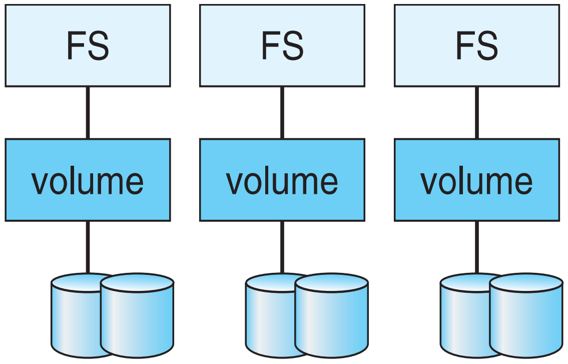
- RAID alone does not prevent or detect data corruption or other errors, just disk failures
- Solaris ZFS adds **checksums** of all data and metadata
- Checksums kept with pointer to object, to detect if object is the right one and whether it changed
- Can detect and correct data and metadata corruption
- ZFS also removes volumes, partitions
  - Disks allocated in **pools**
  - Filesystems with a pool share that pool, use and release space like **`malloc()`** and **`free()`** memory allocate / release calls

ZFS Checksums All Metadata and Data

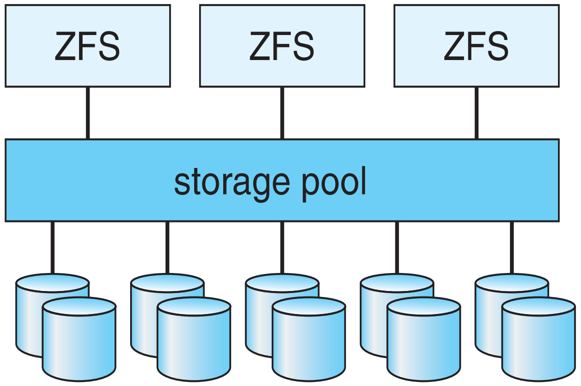


ZFS checksums all metadata and data

Traditional and Pooled Storage



(a) Traditional volumes and file systems.



(b) ZFS and pooled storage.

Object Storage

- General-purpose computing, file systems not sufficient for very large scale
- Another approach – start with a storage pool and place objects in it
  - Object just a container of data
  - No way to navigate the pool to find objects (no directory structures, few services
  - Computer-oriented, not user-oriented
- Typical sequence
  - Create an object within the pool, receive an object ID
  - Access object via that ID
  - Delete object via that ID

Object Storage (Cont.)

- Object storage management software like **Hadoop file system (HDFS)** and **Ceph** determine where to store objects, manages protection
  - Typically by storing N copies, across N systems, in the object storage cluster
  - **Horizontally scalable**
  - **Content addressable, unstructured**

Thanks

Politecnico di Torino
Department of Control and Computer Engineering

Questions?

sarah.azimi@polito.it