CISC 7310X
C11a Mass Storage

Hui Chen

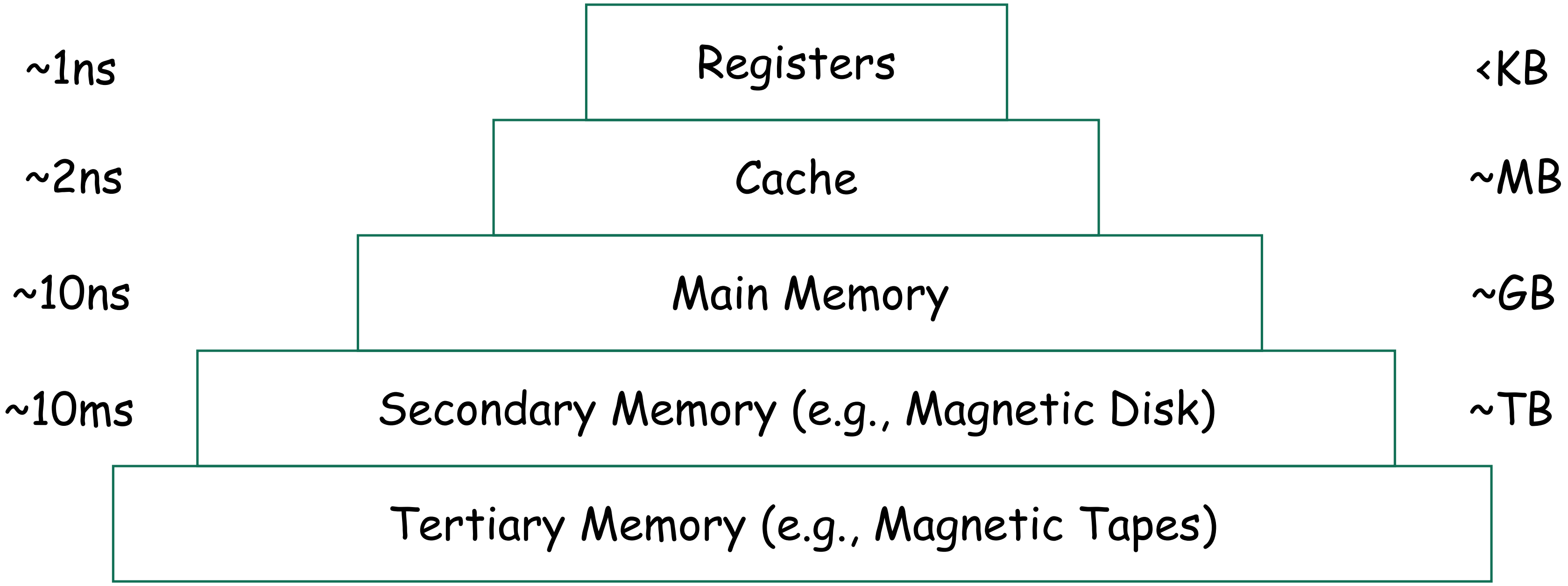
Department of Computer & Information Science

CUNY Brooklyn College

Acknowledgement

- These slides are a revision of the slides provided by the authors of the textbook via the publisher of the textbook

Memory Hierarchy


~1ns
Registers
<KB
~2ns
Cache
~MB
~10ns
Main Memory
~GB
~10ms
Secondary Memory (e.g., Magnetic Disk)
~TB
Tertiary Memory (e.g., Magnetic Tapes)

Outline

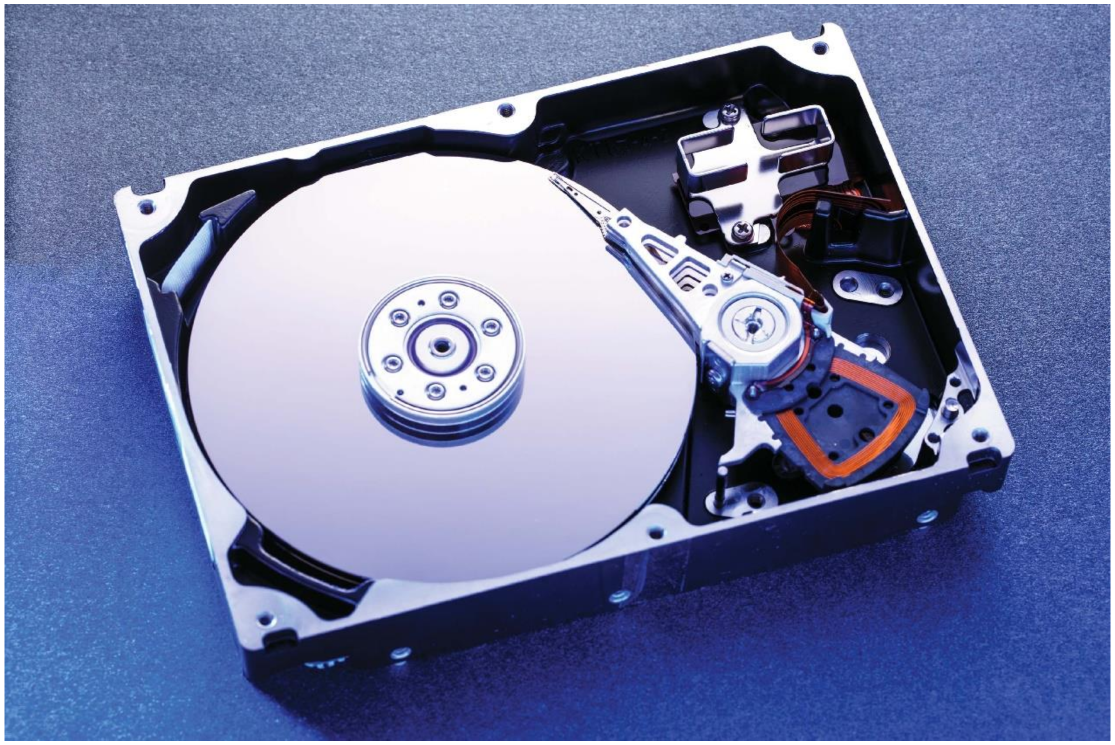
- Overview of Mass Storage Structure
- HDD Scheduling
- NVM Scheduling
- Error Detection and Correction
- Storage Device Management
- Swap-Space Management
- Storage Attachment
- RAID Structure

Overview of Mass Storage Devices

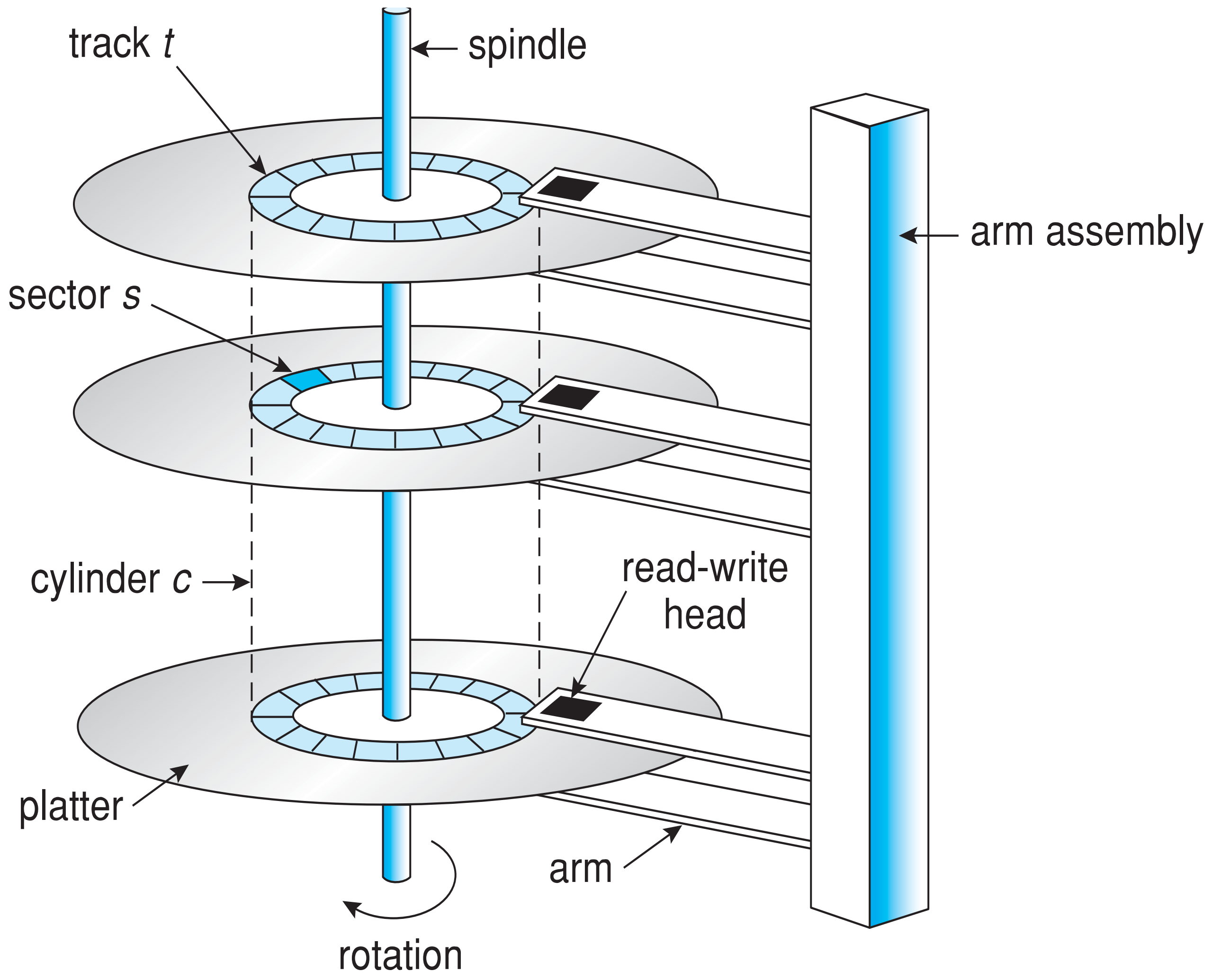
- Bulk of secondary storage for modern computers are
  - Hard disk drives (HDDs)
  - Nonvolatile memory (NVM) devices
- Volatile memory frequently used as a mass storage device
- Magnetic tapes were used as a secondary storage

Hard Disk Drives

- HDDs spin platters of magnetically-coated material under moving read-write heads



Moving-head Disk Mechanism



HDD Performance

- Rotational speed
- Transfer rate
- Positioning time (or random access time)

Rotation Speed

- A disk drive motor spins it at high speed.
- Most drives rotate 60 to 250 times per second
- Common drives spin at 5,400, 7,200, 10,000, and 15,000 Rotations Per Minute (RPM)
- Some drives power down when not in use and spin up upon receiving an I/O request.

Transfer rate

- Transfer rate is rate at which data flow between drive and computer
  - Commonly ~ gigabits / second (Gb/s)

Random Access Time

- Positioning time (or random access time) consists of two parts
  - Seek time
    - time to move disk arm to desired cylinder
  - Rotational latency
    - time for desired sector to rotate under the disk head (rotational latency)

Head Crash

- Head crash results from disk head making contact with the disk surface
  - A head crash normally cannot be repaired

HDD Characteristics: Size & Capacity: Examples

- Platters range from .85” to 14” (historically)
  - Commonly 3.5”, 2.5”, and 1.8”
- Range from ~100GB to ~10TB per drive

HDD Performance: Transfer Rate: Examples

- Transfer Rate
  - Theoretical/stated, 6 Gb/sec
- Effective Transfer Rate
  - Real/can be achieved/actual, ~1Gb/sec

HDD Performance: Random Access Time: Examples

- Seek time from 3ms to 12ms – 9ms common for desktop drives
  - Average seek time measured or calculated for movement between tracks
- Rotational latency based on spindle speed
  - 1 / (RPM / 60) = 60 / RPM
  - Average rotational latency = $\frac{1}{2}$ rotational latency

HDD Performance: Access Latency: Examples

- **Access latency** = **Average access time** = average seek time + average (rotational) latency
    - Examples
        - For fastest disk 3ms + 2ms = 5ms
        - For slow disk 9ms + 5.56ms = 14.56ms

Average I/O Time: Example

- Transfer a 4KB block on a 7200 RPM disk with a 5ms average seek time, 1Gb/sec transfer rate with a .1ms controller overhead
  - Average access time = average seek time + average latency = 5 + ½ 1/(7200/60) 1000 ≈ 5 ms + 4.17 ms
  - Transfer time = 4KB / 1Gb/s ≈ 0.031 ms
  - Average I/O time for 4KB block ≈ (5ms + 4.17ms) + 0.1ms + 0.031 ms = 9.301 ms
- Which part is dominant?

Issues with HDD

- Seek time is dominant
- How to minimize seek time?
    - Disk scheduling/Disk arm scheduling algorithms

Questions?

- Hard disk drives
- Characteristics and performance?

Nonvolatile Memory Devices

NVM Devices Characteristics

- Have different forms
  - If disk-drive like, then called **solid-state disks** (**SSD**s)
  - Other forms include **USB drives** (thumb drive, flash drive), DRAM disk replacements, surface-mounted on motherboards, and main storage in devices like smartphones
- Can be more reliable than HDDs
- Maybe have shorter life span – need careful management
- Less capacity
- But much faster
- Buses can be too slow -> connect directly to PCI for example
- No moving parts, so no seek time or rotational latency

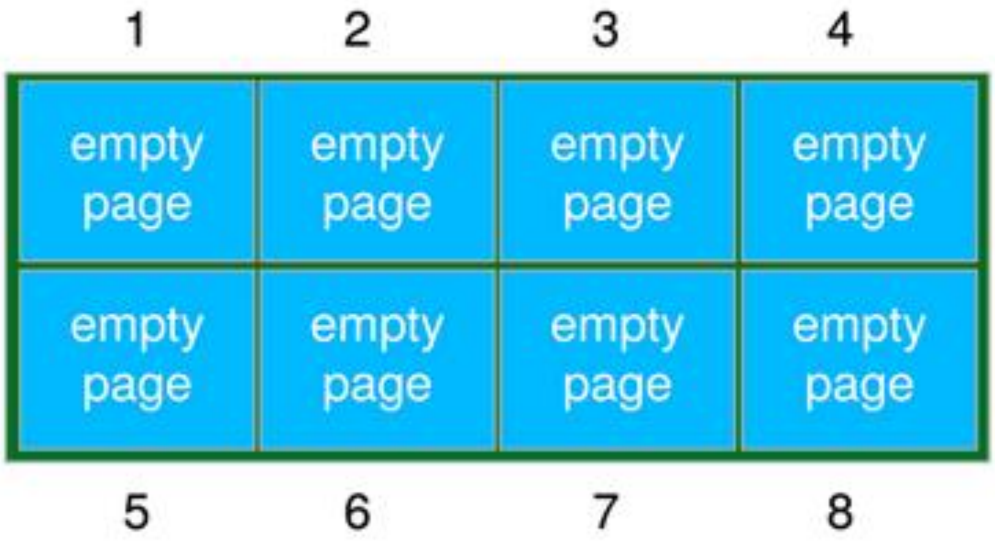
Issues with NVM Devices

- 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" increments
  - 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

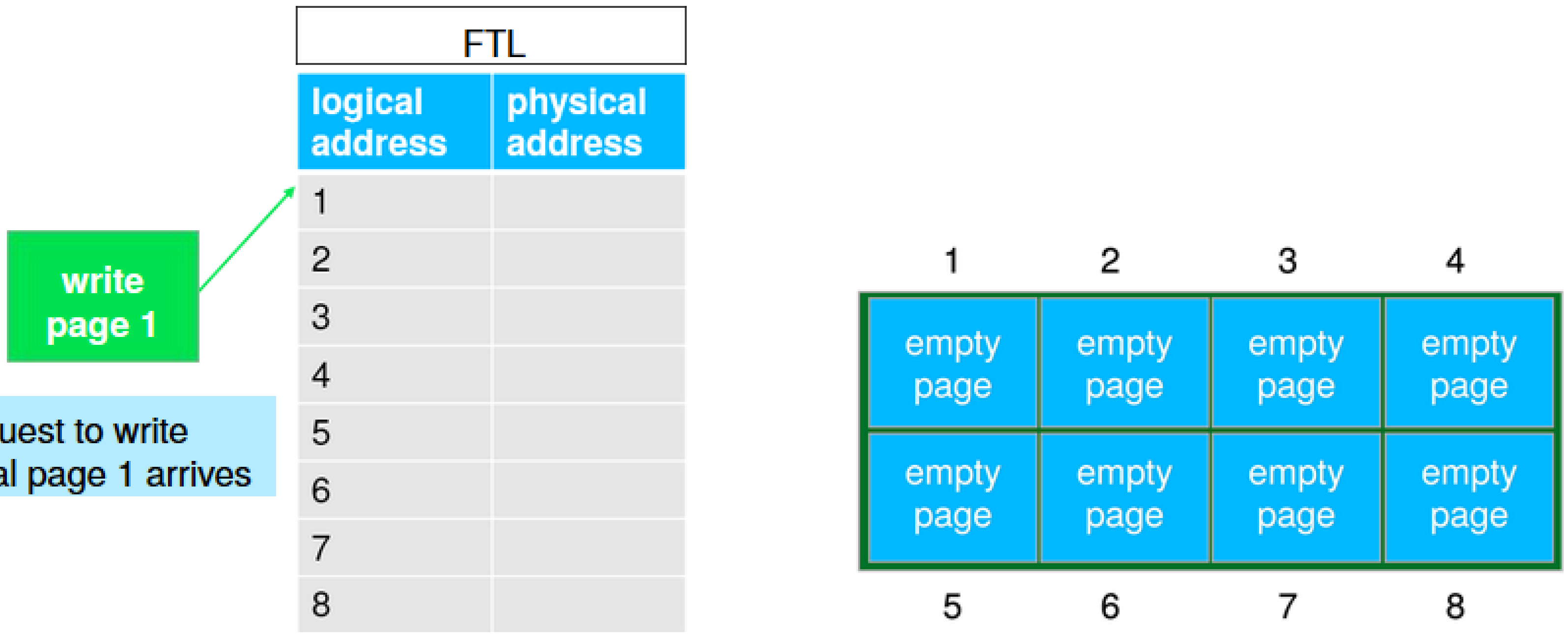
NAND Flash Controller Algorithms

- 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
- Allocates **overprovisioning** to provide working space for GC
- Each cell has lifespan, so **wear leveling** needed to write equally to all cells

FTL	
logical address	**physical address**
1	
2	
3	
4	
5	
6	
7	
8	



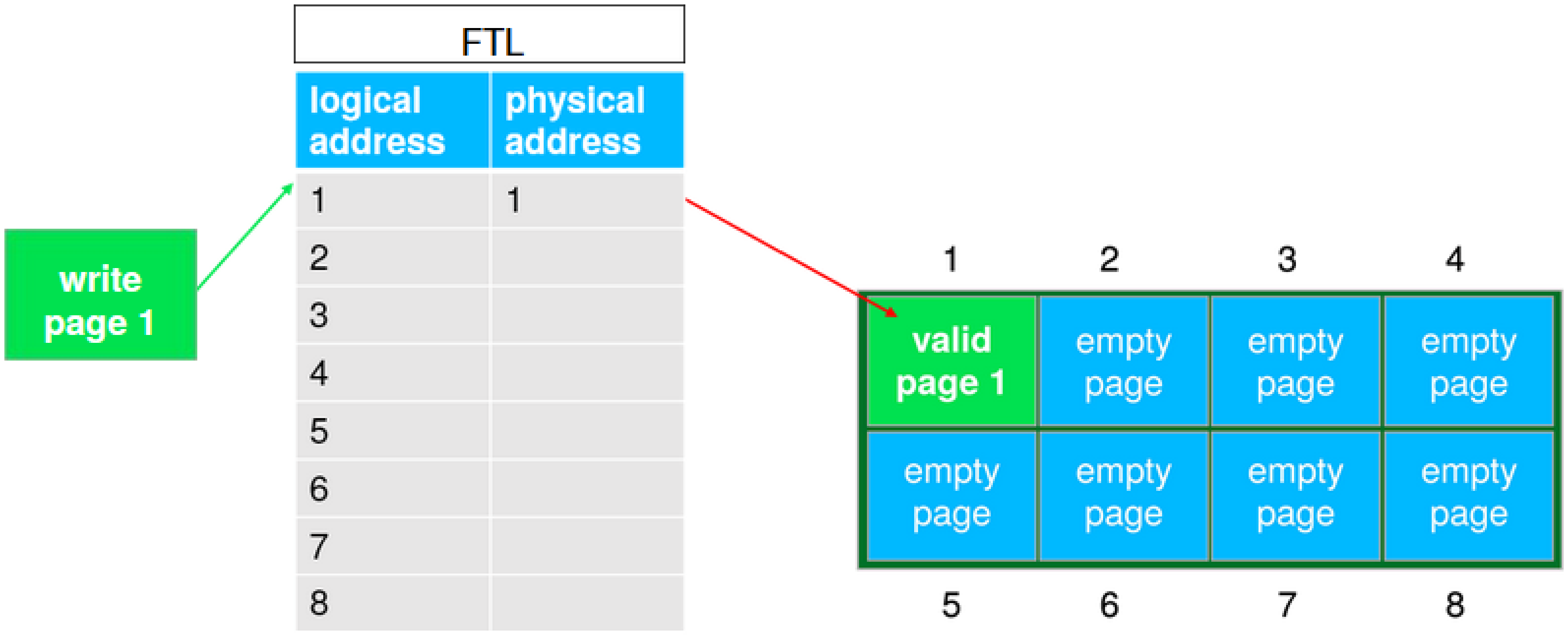
1. assume eight pages in one block of NAND flash storage, not yet written to



FTL

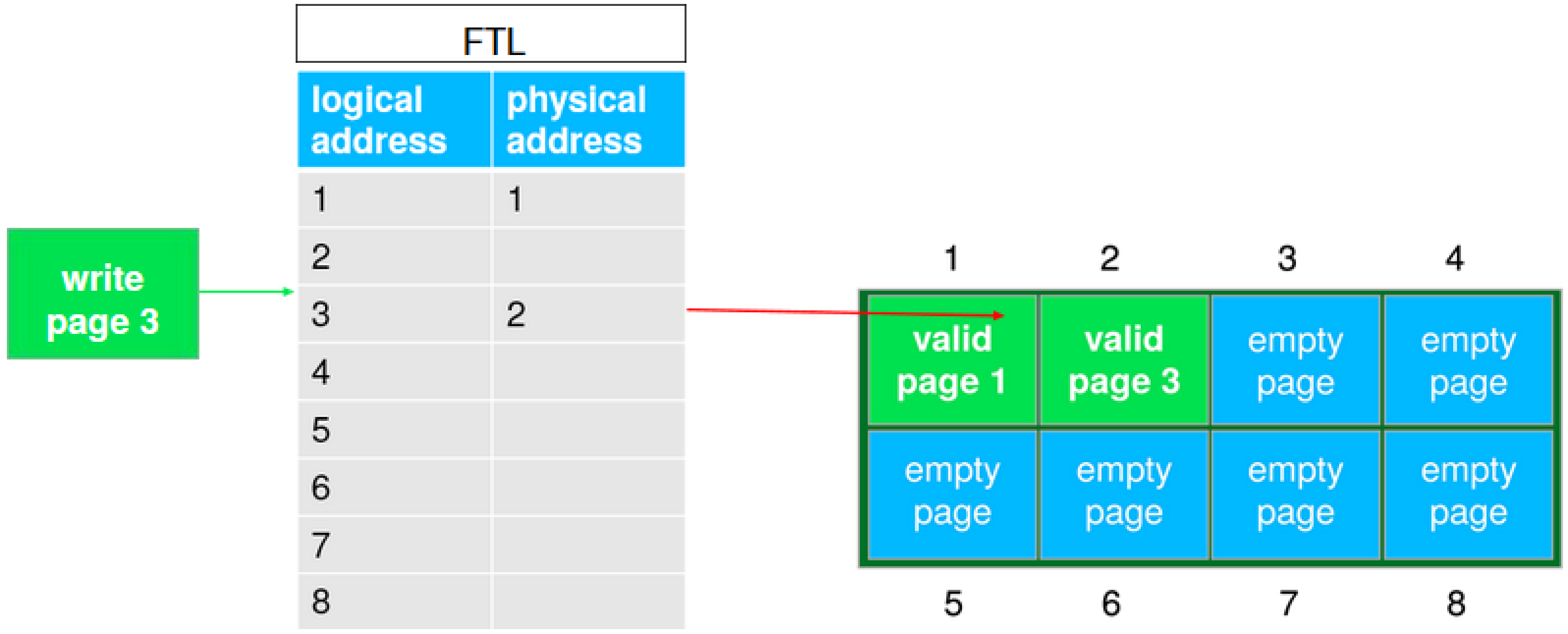
logical address	physical address
1	
2	
3	
4	
5	
6	
7	
8	

2. a request to write logical page 1 arrives



logical address	physical address
1	1
2	
3	
4	
5	
6	
7	
8	

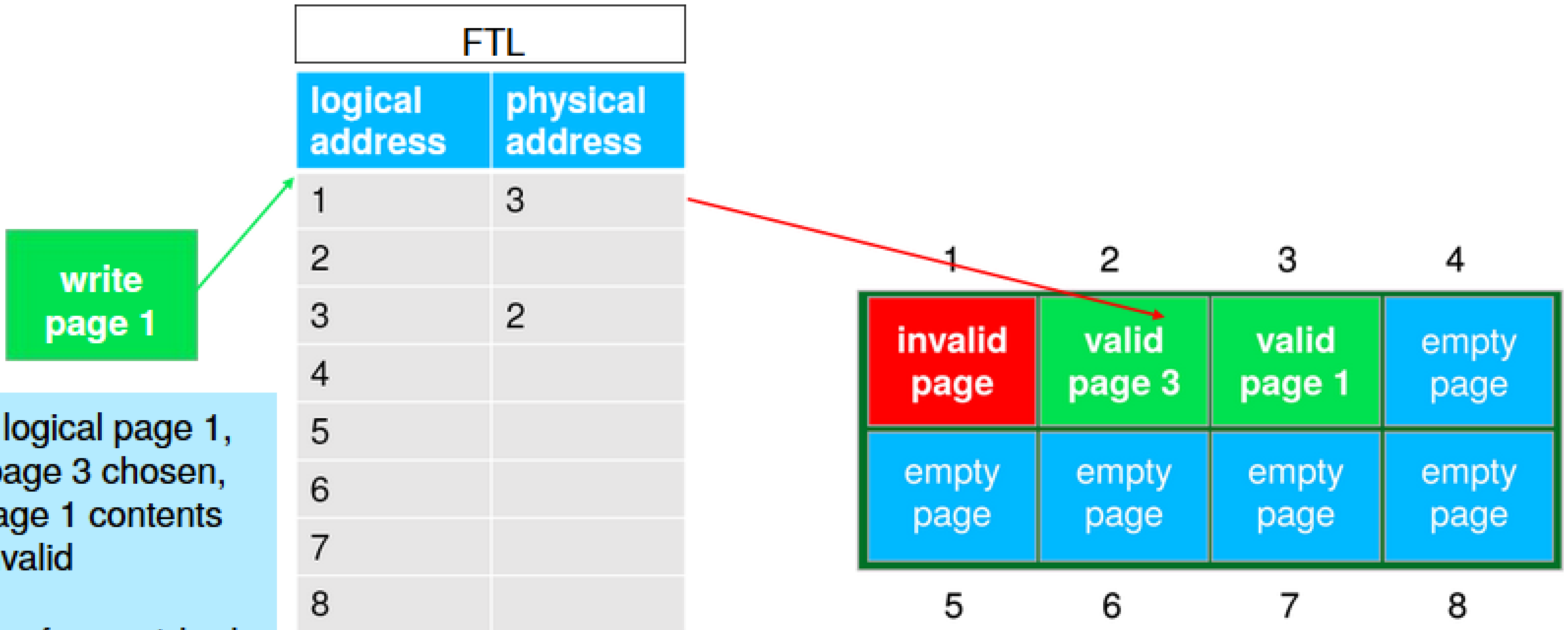
3. physical page 1 chosen, logical page 1 written to flash physical page 1



FTL

logical address	physical address
1	1
2	
3	2
4	
5	
6	
7	
8	

4. logical page 3 write request, physical page 2 chosen and written to



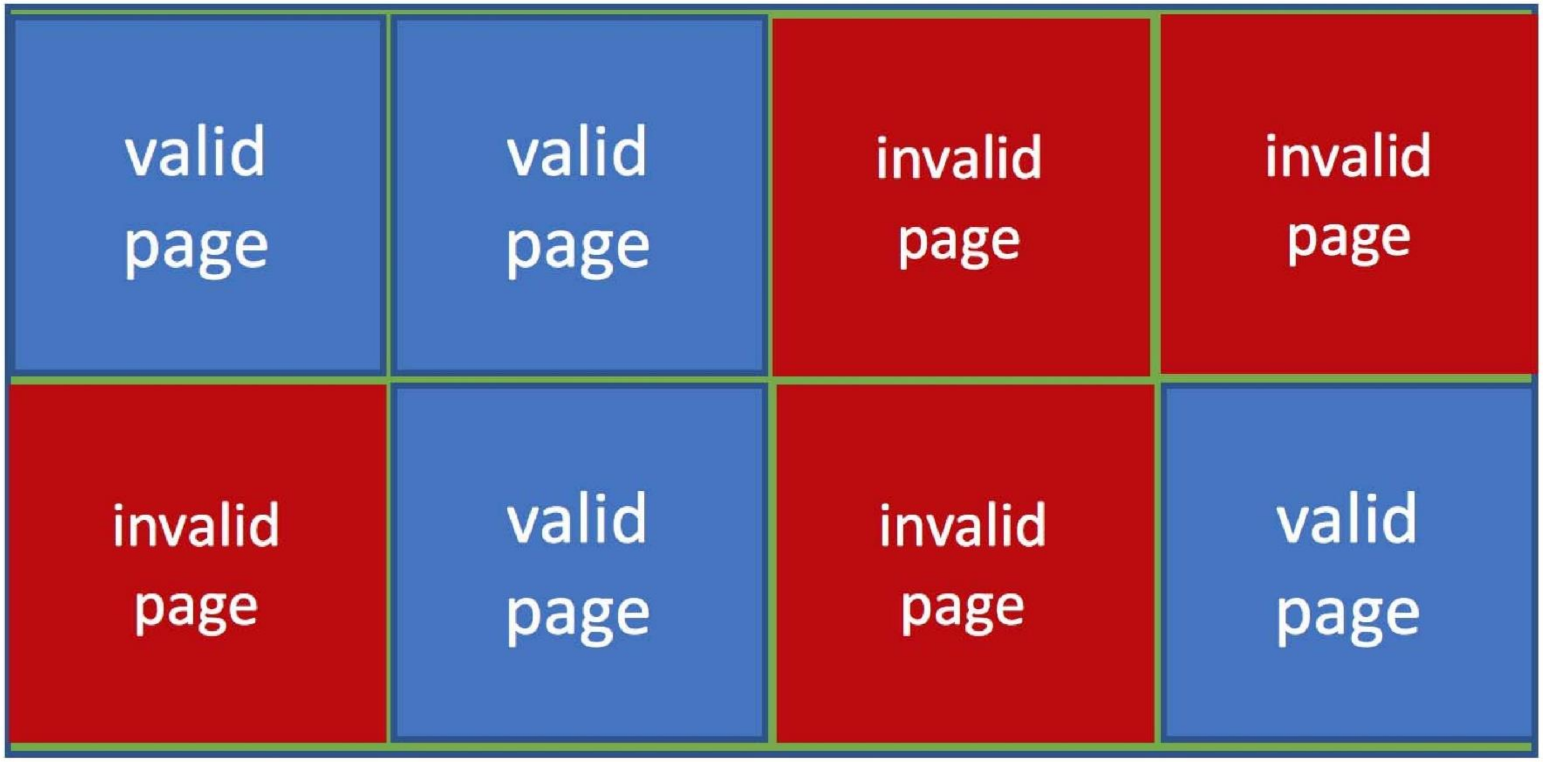
FTL

logical address	physical address
1	3
2	
3	2
4	
5	
6	
7	
8	

5. update to logical page 1, physical page 3 chosen, original page 1 contents marked invalid

   if only part of page 1 had been updated, part of physical page 1 would have been marked invalid and data would have been garbage collected

NAND block with Valid and Invalid pages: Example



NAND block with valid and invalid pages

Questions?

- NVM
- Issues with NVM

Volatile Memory

- DRAM frequently used as mass-storage device
  - Not technically secondary storage because volatile, but can have file systems, be used like very fast secondary storage
- **RAM drives** (with many names, including RAM disks) present as raw block devices, commonly file system formatted

Why RAM Disks?

- 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

RAM Disk Example

- On Linux

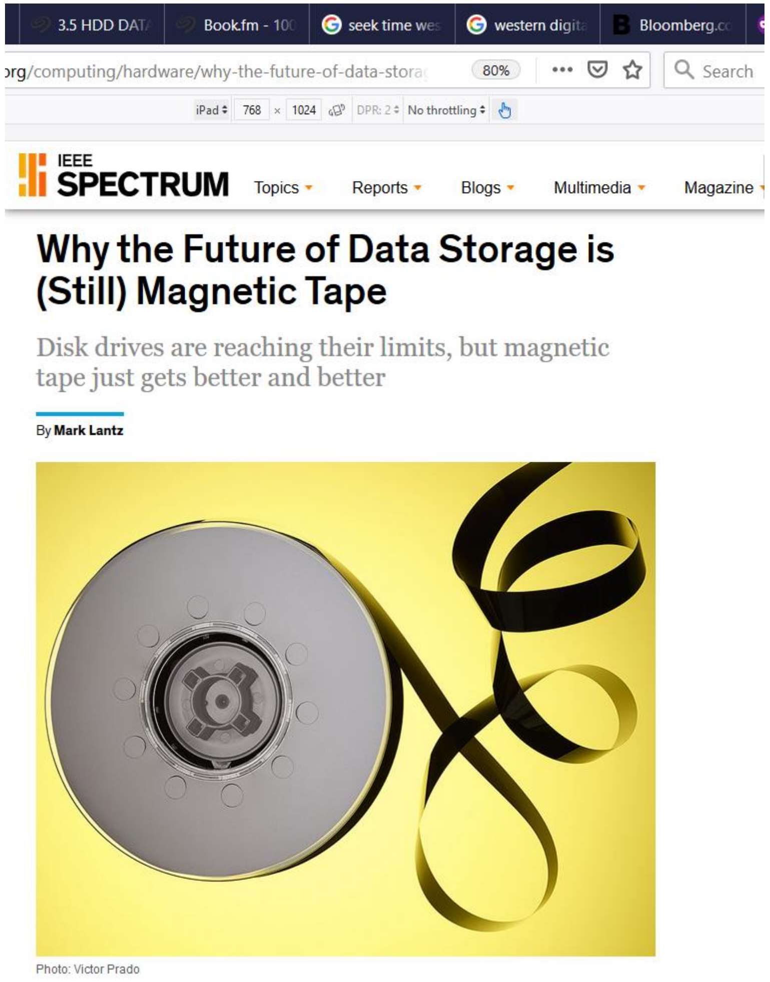
```
mkdir ramdisk
mount -t tmpfs -o size=5m tmpfs ramdisk
```

Magnetic Tape

- Nonvolatile, hold large quantities of data
- Used as an early secondary-storage medium.
- Its access time is slow compared with that of main memory and drives.
- Serve as backup/tertiary storage nowadays

Use of Magnetic Tape: Example

- https://www.youtube.com/watch?v=avP5d16wEp0#t=01m09s



Why the Future of Data Storage is (Still) Magnetic Tape

Disk drives are reaching their limits, but magnetic tape just gets better and better

By **Mark Lantz**

Photo: Victor Prado

Questions

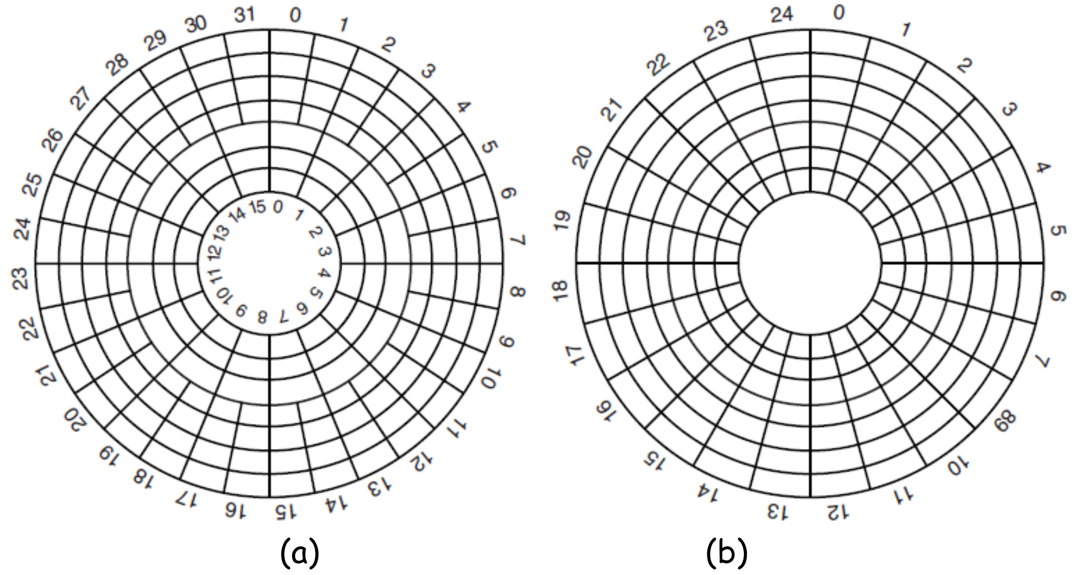
- Characteristics, performances, and issues of HDD
- Characteristics, performances, and issues of NVM
- Characteristics, performances, and issues of RAM Disks
- Characteristics, performances, and issues of magnetic tapes

Disk Structure

- Logical blocks
  - To the host, the disks are a one-dimensional array of logical blocks
  - Smallest unit of data transfer between the disk and the host
- Disk sector
  - A logical block mapped to one or more disk sectors
  - A sector is typically 2^9 = 512 bytes
- Constant linear velocity
  - Optical disks
- Constant angular velocity
  - Magnetic disks

Logical and Physical Geometry

- Logical (virtual) geometry and physical geometry are differ
  - Traditionally, (x, y, z): (cylinders, heads, sectors), i.e., CHS
    - PC: (65535, 16, 63), a sector is typically 512 bytes



- [Figure 5-19 in Tanenbaum & Bos, 2014] Figure 5-19. (a) Physical geometry of a disk with two zones. (b) A possible virtual geometry for this disk

Disk Structure

- Logical (virtual) geometry and physical geometry are differ
  - Traditionally, (x, y, z): (cylinders, heads, sectors), i.e., CHS
    - PC: (65535, 16, 63), a sector is typically 512 bytes
  - Modern approach: logical block addressing (LBA), disk sectors numbered consecutively starting at 0

Logical Block Addressing

- Disk drives are addressed as large 1-dimensional arrays of logical blocks, where the logical block is the smallest unit of transfer
  - Low-level formatting creates logical blocks on physical media
- 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 for bad sectors
    - Non-constant # of sectors per track via constant angular velocity

Questions?

- HDD structure and addressing

Using HDD Efficiently

- The operating system is responsible for using hardware efficiently
- For the disk drives, this means having a fast access time and disk bandwidth
- Access time = seek time + rotational latency + data transfer time + controller overhead
- Disk **bandwidth**
    - (the total number of bytes transferred)/(the total time between the first request for service and the completion of the last transfer)

Seek Time

- Recall the factors of a disk block read/write performance
  - Seek time (the time to move the arm to the proper cylinder)
  - Rotational latency (how long for the proper sector to come under the head)
  - Data transfer time
  - Controller overhead
- The seek time often dominates.
- Can we reduce the seek time?
- Seek time $\approx$ seek distance, can we reduce seek distance?
- When seek time becomes less, data bandwidth becomes greater

Disk I/O Request Queue and Disk Scheduling

- There are many sources of disk I/O request
    - OS, system processes, and users processes
    - I/O request includes
        - input or output mode, disk address, memory address, number of sectors to transfer
- OS maintains queue of requests, per disk or device
- Idle disk can immediately work on I/O request, busy disk means work must queue
- Optimization algorithms only make sense when a queue exists
- Optimization is to reduce seek distance → Disk scheduling/disk arm scheduling algorithms

Where is it?

- In the past, operating system responsible for queue management, disk drive head scheduling
- Now, built into the storage devices, controllers
  - Just provide LBAs, handle sorting of requests
- To discuss some of a few algorithms they use

Disk Scheduling Problem

- Assumption
  - A disk driver accepts bursts of access requests
- Problem
  - In which order should these requests be carried out to minimize the seek time?
    - Average seek time?
    - Overall seek time?
- Algorithms
  - FCFS
  - SCAN
  - C-SCAN

Example Requests for Analysis

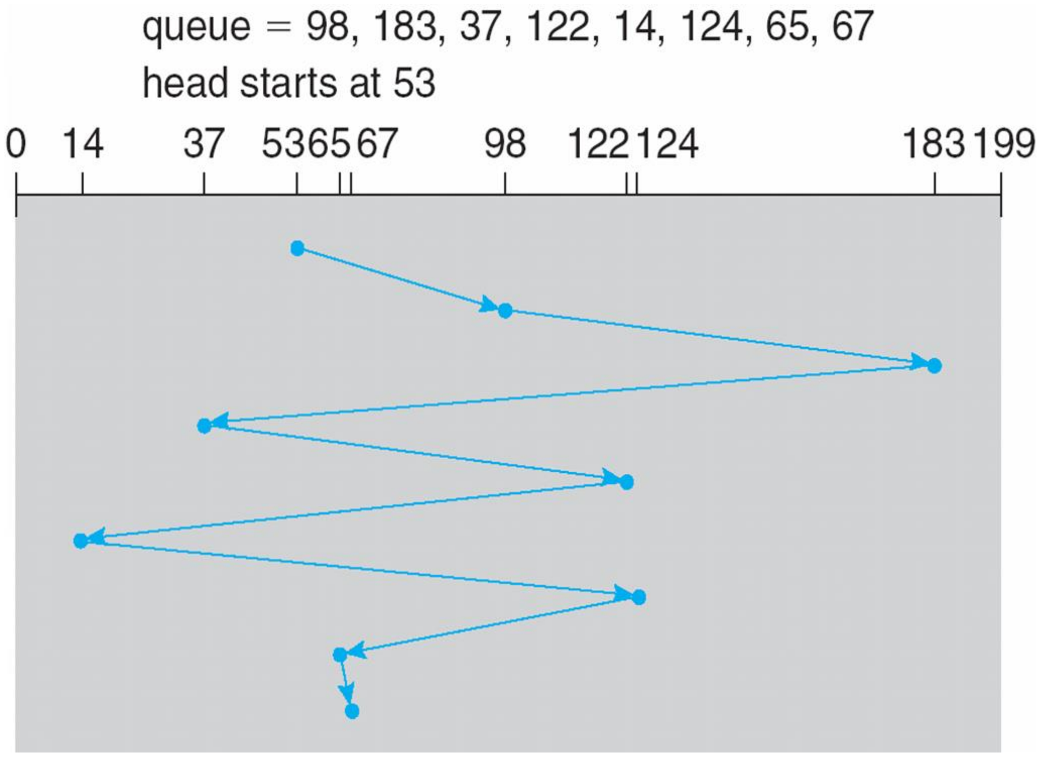
- A request queue (0-199): a list of HDD cylinder numbers

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

- Head pointer 53 at the beginning

FCFS

- Illustration shows total head movement of 640 cylinders

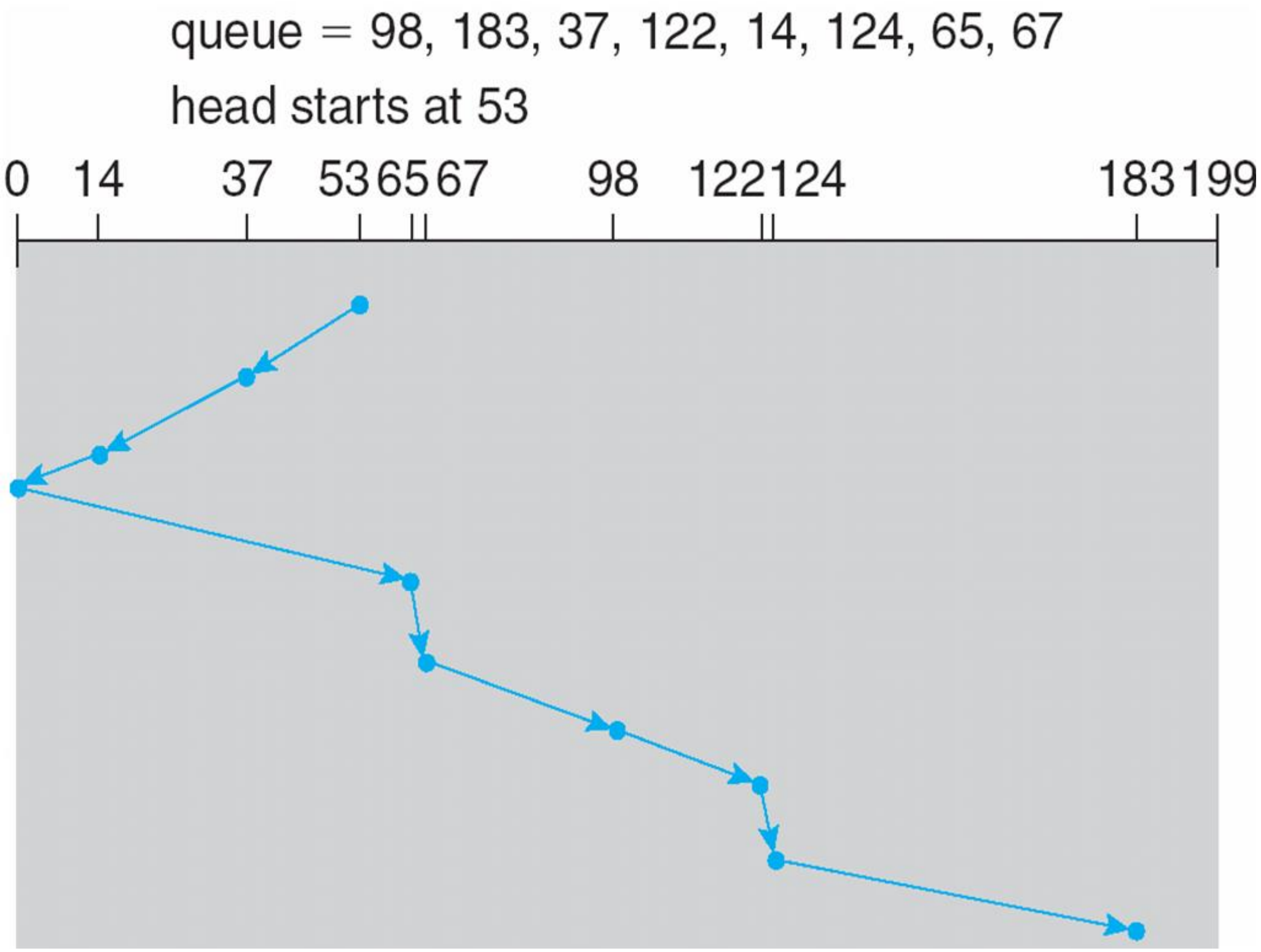


SCAN

- **SCAN algorithm** Sometimes called the **elevator algorithm**
  - The disk arm starts at one end of the disk and moves toward the other end, servicing requests as it reaches each cylinder, until it gets to the other end of the disk.
  - At the other end, the direction of head movement is reversed, and servicing continues.
  - The head continuously scans back and forth across the disk..

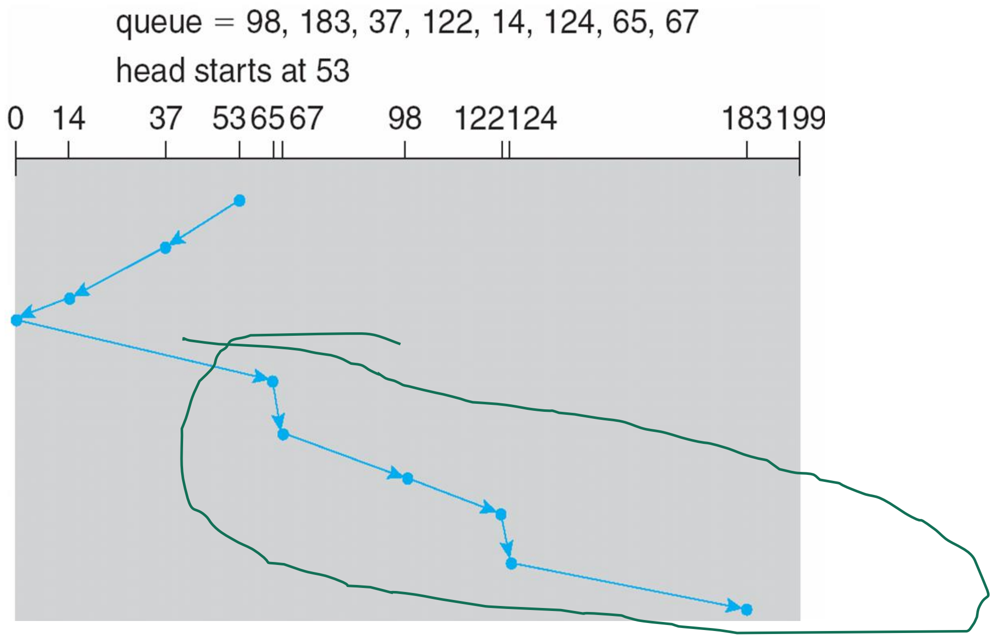
SCAN: Example

- Illustration shows total head movement of 208 cylinders


queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
0
14
37
53
65
67
98
122
124
183
199

SCAN: Observation

- But note that if requests are uniformly dense, largest density at other end of disk and those wait the longest

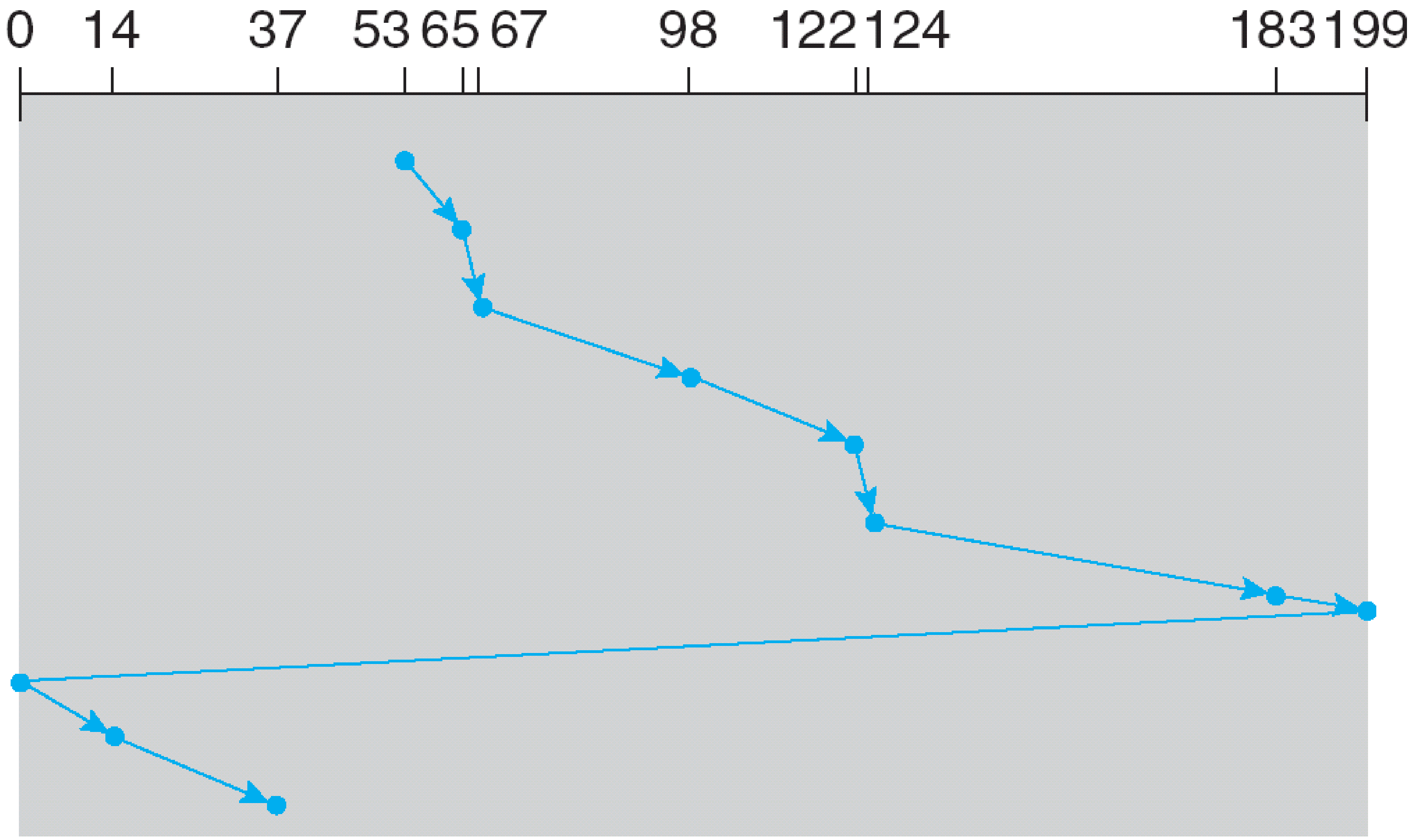


Circular-Scan (C-SCAN)

- 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
- Provides a more uniform wait time than SCAN

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

head starts at 53



Selecting a Disk-Scheduling Algorithm

- SCAN and C-SCAN perform better for systems that place a heavy load on the disk
- 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
  - 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
- In RHEL 7 also NOOP and completely fair queueing scheduler (CFQ) also available, defaults vary by storage device
  - NOOP for CPU-bound systems using fast storage such as NVM devices,
  - CFA for SATA HDDs

Questions?

- Disk scheduling/Disk arm scheduling
- FCFS
- SCAN
- C-SCAN
- Which one to use?

NVM Scheduling

- No disk heads or rotational latency but still room for optimization
- The observed behavior of NVM devices
  - the time required to service reads is uniform, but
  - write service time is not uniform.
  - Some SSD schedulers have exploited this property and merge only adjacent write requests, servicing all read requests in FCFS order.
- 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

Questions?

- Can we optimize NVM?
  - NVM Scheduling?

Error Detection and Correction

- Fundamental aspect of many parts of computing (memory, networking, storage)
- Error detection determines if there a problem has occurred (for example a bit flipping)
    - If detected, can halt the operation
    - Detection frequently done via parity bit
- Parity 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

Questions?

- Questions about error detection and correction?

Storage Device Management

- Drive Formatting, Partitions, and Volumes
  - Low-level formatting
  - Volume creation and management
  - Partition and logical formatting
  - Boot block

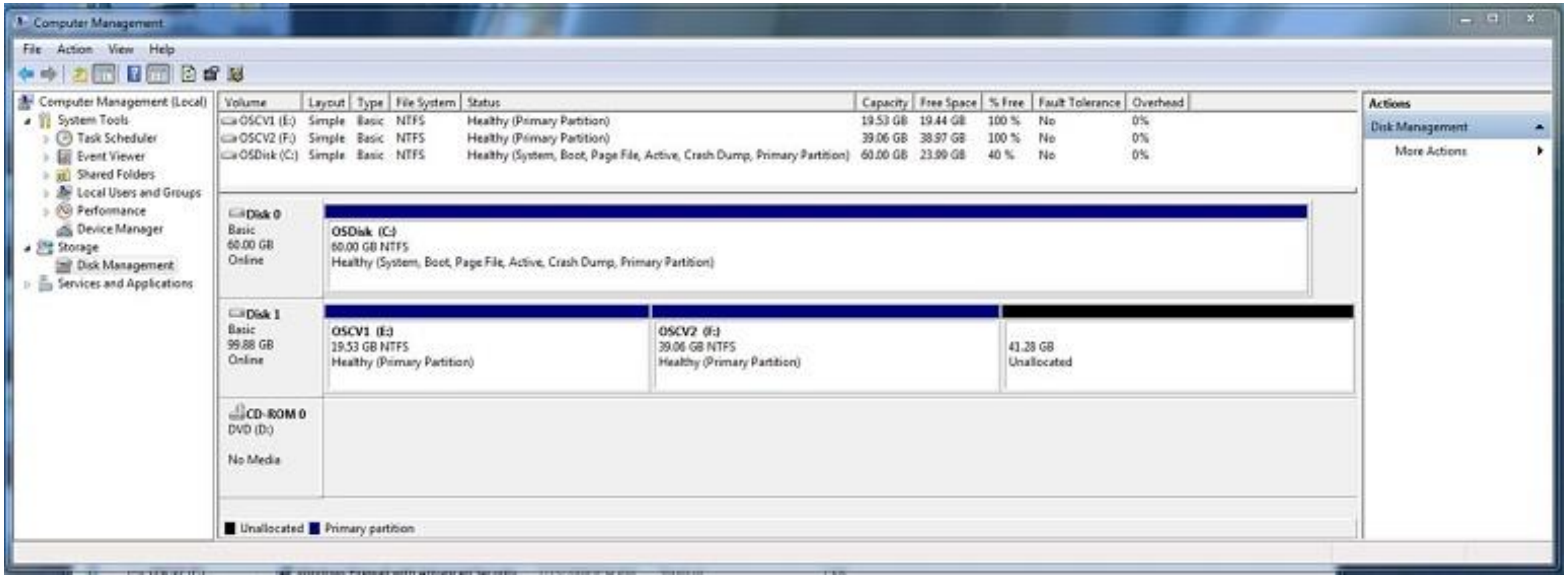
Drive Formatting

- **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, plus data, plus error correction code (**ECC**)
  - Usually 512 bytes of data but can be selectable

Partition and Logical Formatting

- 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 cylinders, each treated as a logical disk
- Logical formatting or "making a file system"
- To increase efficiency most file systems group blocks into clusters
    - Disk I/O done in blocks
    - File I/O done in clusters

Windows



Volume	Layout	Type	File System	Status	Capacity	Free Space	% Free	Fault Tolerance	Overhead
OSCV1 (E:)	Simple	Basic	NTFS	Healthy (Primary Partition)	19.53 GB	19.44 GB	100 %	No	0%
OSCV2 (F:)	Simple	Basic	NTFS	Healthy (Primary Partition)	39.06 GB	38.97 GB	100 %	No	0%
OSDisk (C:)	Simple	Basic	NTFS	Healthy (System, Boot, Page File, Active, Crash Dump, Primary Partition)	60.00 GB	23.99 GB	40 %	No	0%

Disk 0
Basic
60.00 GB
Online

OSDisk (C:)
60.00 GB NTFS
Healthy (System, Boot, Page File, Active, Crash Dump, Primary Partition)

Disk 1
Basic
99.88 GB
Online

OSCV1 (E:)
19.53 GB NTFS
Healthy (Primary Partition)

OSCV2 (F:)
39.06 GB NTFS
Healthy (Primary Partition)

41.28 GB
Unallocated

CD-ROM 0
DVD (D:)

No Media

■ Unallocated ■ Primary partition

Windows Disk Management tool showing devices, partitions, volumes, and file systems

Linux

```
# cat /proc/diskstats
# fdisk –l /dev/sda
# sfdisk -d /dev/sda
```

Root and Non-Root Partitions

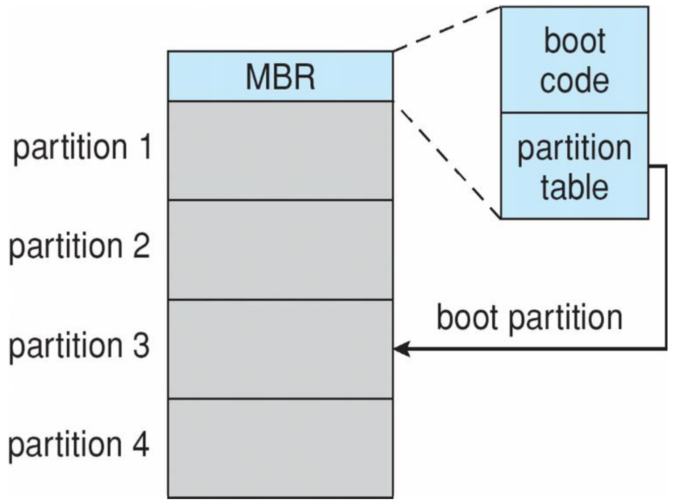
- Root partition contains the OS
  - Mounted at boot time
    - At mount time, file system consistency checked
    - Is all metadata correct?
      - If not, fix it, try again
      - If yes, add to mount table, allow access
- Other partitions can hold other OSes, other file systems, or be raw
  - Other partitions can mount automatically or manually

Boot Block

- 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 point to a boot management program for multi-OS booting

Boot Block Initialization

- Boot block initializes system
    - The bootstrap is stored in ROM, firmware
    - **Bootstrap loader** program stored in boot blocks of boot partition



Booting from secondary storage in Windows

Linux Grub MBR

- On a Debian Linux system, assume 32bit architecture,

```
# apt-get install nasm
# cat /proc/diskstats
# dd if=/dev/sda of=mbr.bin bs=512 count=1
# hextedit mbr.bin
# ndisasm -b16 -o7c00h mbr.bin > mbr.asm
# vi mbr.asm
```

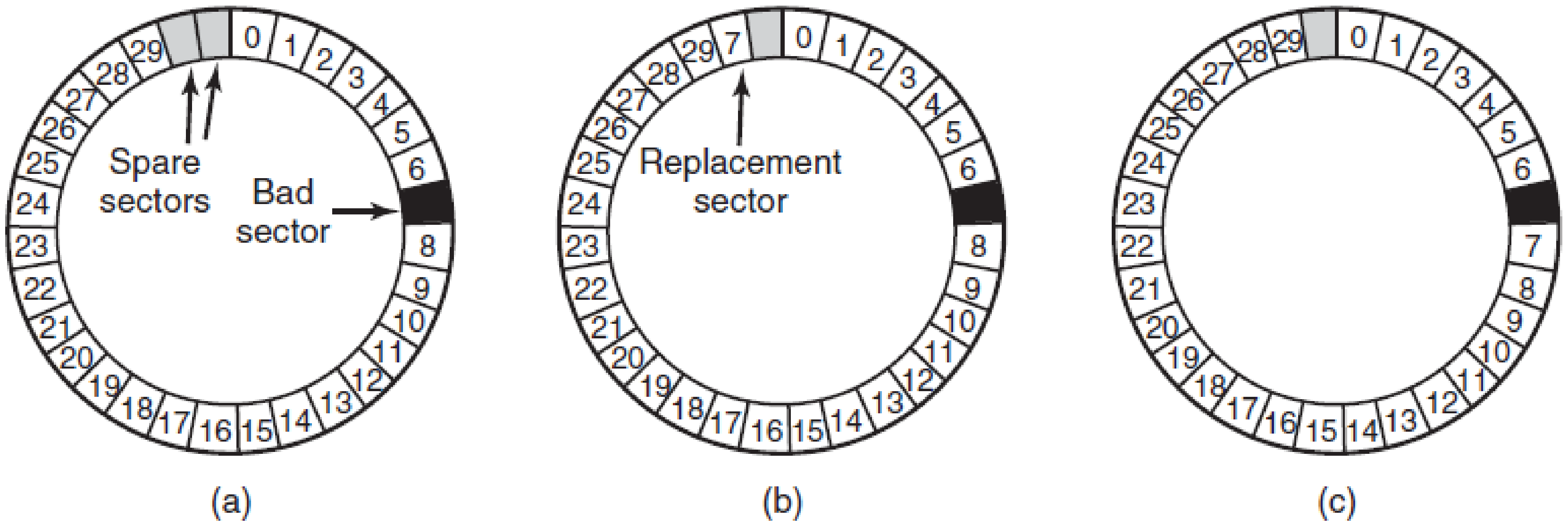
- Partition table starts at offset 446
- For more see https://thestarman.pcministry.com/asm/mbr/GRUB.htm

Bad Blocks

- Methods such as **sector sparing** used to handle bad blocks
  - A typical bad-sector transaction might be as follows:
    1. The operating system tries to read logical block 87.
    2. The controller calculates the ECC and finds that the sector is bad. It reports this finding to the operating system as an I/O error.
    3. The device controller replaces the bad sector with a spare.
    4. After that, whenever the system requests logical block 87, the request is translated into the replacement sector's address by the controller.
- Alternatively, some controllers can be instructed to replace a bad block by **sector slipping**

Spare Substitution

- Two approaches shown in (b) and (c)



- [Figure 5-26 in Tanenbaum & Bos, 2014]

Raw Disk

- Raw disk access for apps that want to do their own block management
  - Keep OS out of the way
  - Example: databases

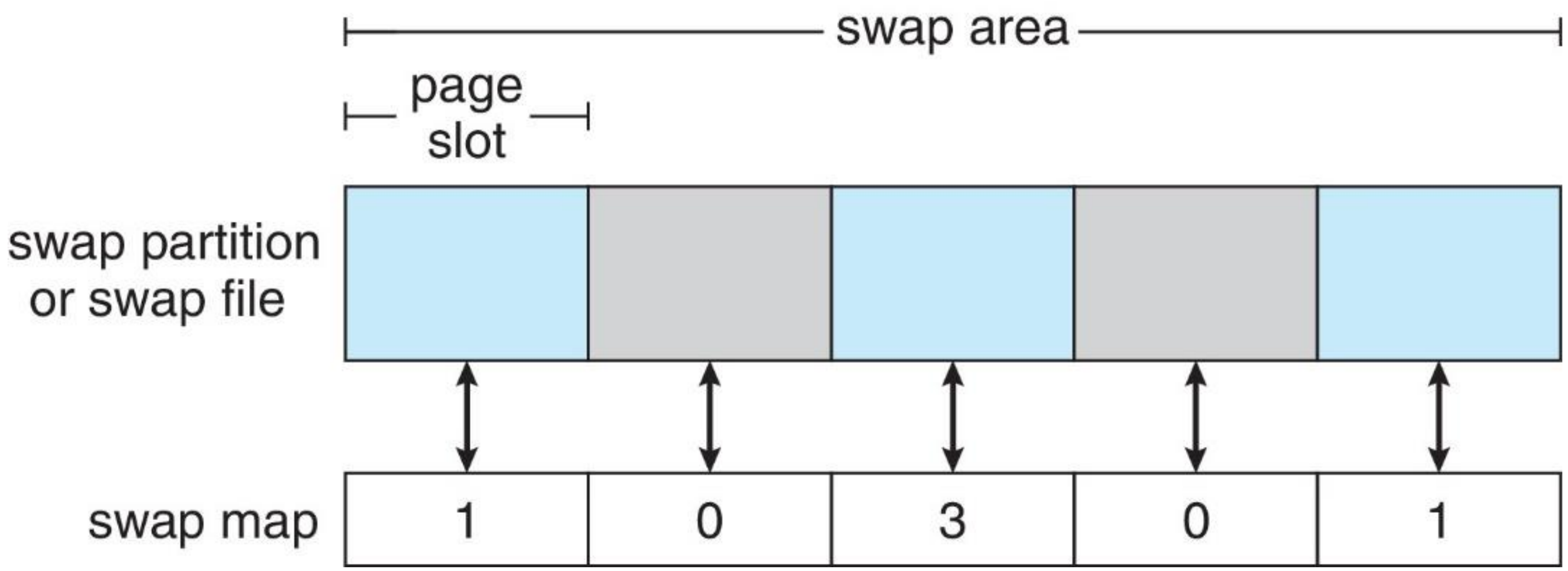
Questions?

- Drive Formatting, Partitions, and Volumes
  - Low-level formatting
  - Volume creation and management
  - Partition and logical formatting
  - Boot block
  - Raw disk

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
  - 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



Questions?

- Swap-space management?

Storage Attachment

- Computers access storage in three ways
  - host-attached
  - network-attached
  - Cloud
- Storage arrays
- Storage array network

Host-Attached Storage

- Host-attached storage accessed through I/O ports talking to I/O buses
- Several busses available, including
  - advanced technology attachment (ATA),
  - serial ATA (SATA, most common)
  - eSATA,
  - serial attached SCSI (SAS),
  - universal serial bus (USB),
  - thunderbolt, and
  - fibre channel (FC), for high-end systems
    - serial architecture using fibre or copper cables
    - Multiple hosts and storage devices can connect to the FC fabric

NVMe

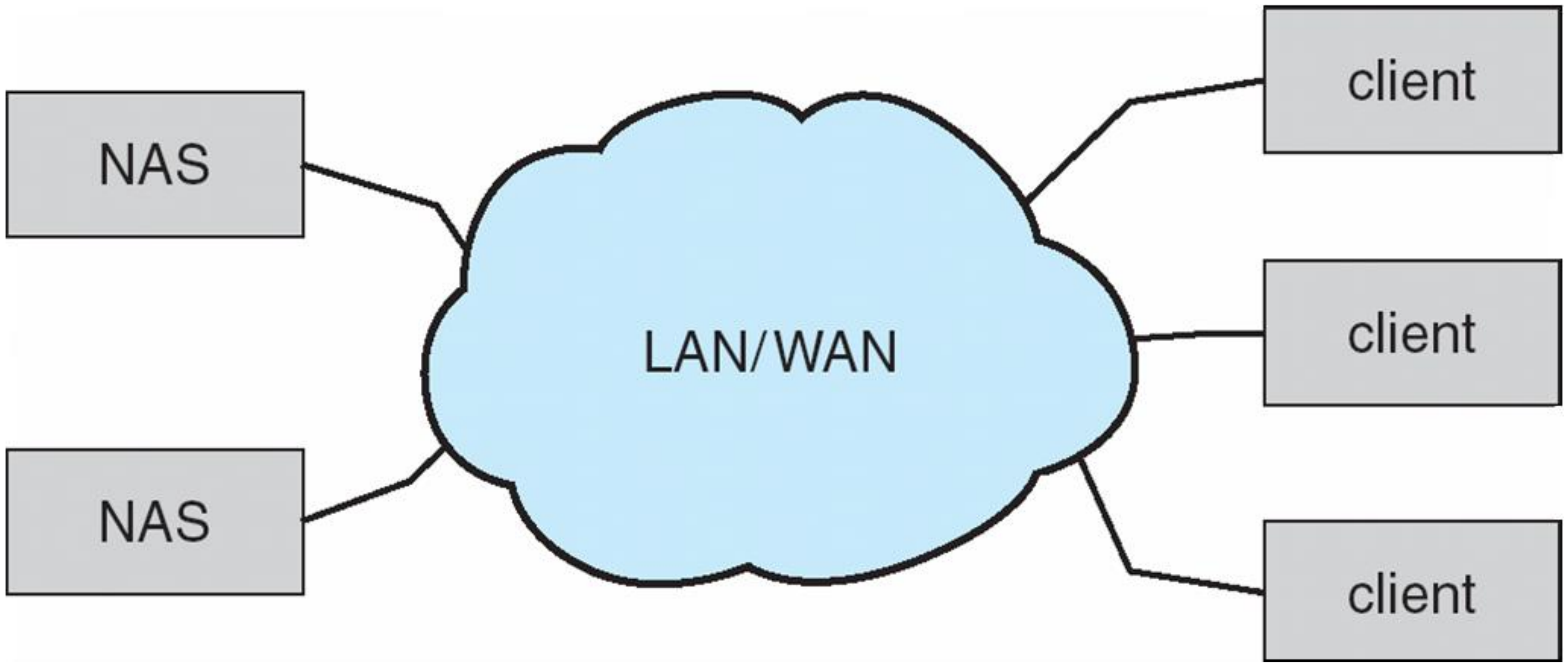
- NVM much faster than HDD
  - New fast interface for NVM called **NVM express** (**NVMe**), connecting directly to PCI bus

Host-Bus Adapters (HBAs)

- 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

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)


NAS
NAS
LAN/WAN
client
client
client

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)

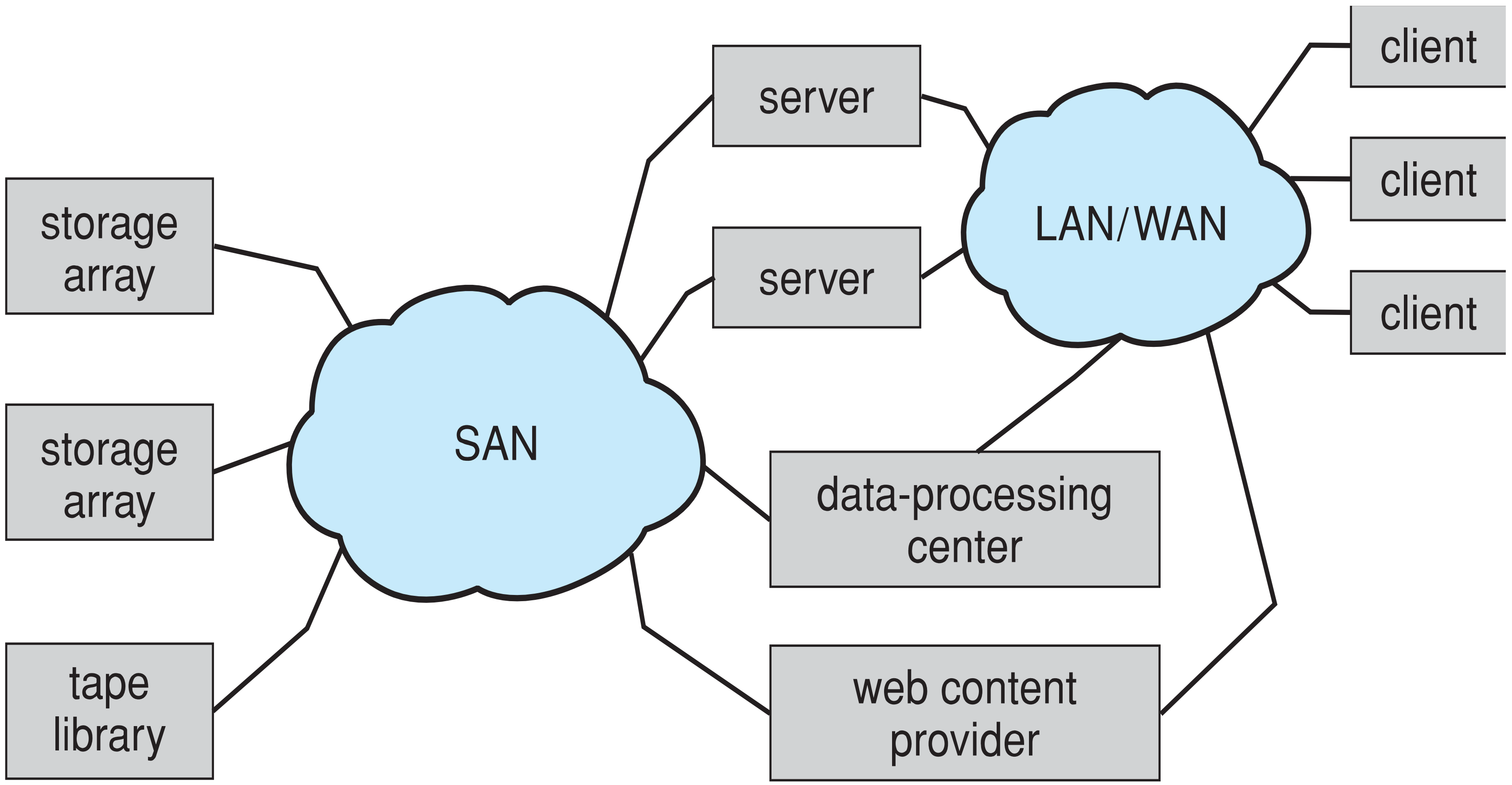
Storage Array

- 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

A Storage Array

Storage Area Network

- Common in large storage environments
- Multiple hosts attached to multiple 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, Fibre Channel (FC), Fibre Channel over Ethernet (FCOE), InifiniBand (IB)


storage array
storage array
tape library
SAN
server
server
LAN/WAN
client
client
client
data-processing center
web content provider

Questions?

- Computers access storage in three ways
  - host-attached
  - network-attached
  - Cloud
- Storage arrays
- Storage array network

Design Goals

- Function
    - They need to work, read & write
- Reliability
    - Murphy’s law
        - "Anything that can go wrong will go wrong“
    - How do we make it appearing reliable?
- I/O Efficiency (performance)
    - We need it to be “fast”
- Energy efficiency

Disk Failures and Data Loss

- Mean time between failures (MTBF)
  - The statistical mean time that a device is expected to work correctly before failing, see an example.
- Mean time to repair
  - Exposure time when another failure could cause data loss
- Mean time to data loss based on above factors

Redundancy and Data Loss

- Failure = lost of data; [Mean time between failure (MTBF)]() of a single disk drive and many disk drives
    - Example
        - MTBF of a single disk drive: 1,000,000 hours
        - 100 disk drives: 1,000,000/100 = 10,000 hours = 416.7 days
- Mirroring: duplicate disk drive
    - Example: two physical disk drives are presented as a logical disk drive (mirrored volume)
    - Failure: failure of two physical disk drives
- Mean time to repair (MTTR): time required to replace the failure disk drive
- With mirroring
    - MTBF = 1,000,000 hours; MTTR = 10 hours
    - Mean time to data loss: failure of the mirrored volume (the 2nd disk drive also failed before the 1st could be replaced) : $1{,}000{,}000^2$ / (2 x 10) =5×10^{10} hours = 5.7×10^6 years!

Practical Consideration

- Disk failures are not independent
- Example
  - Large number of disk drive failures can be the result of a power failure and a tremor of a minor earthquake
  - Manufacturing defects in a batch of disk drives can lead to correlated failures
  - The probability of failure grows as disk drives age
- Mean time to data loss is smaller

Parallelism and Performance

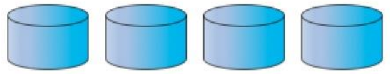
- Observation
  - Duplicating disk drives doubles the rate at which read requests can be handled
- Data stripping
  - Splitting data across multiple disk drives
  - Bit-level stripping
    - Splitting the bits of each byte across multiple disk drives
      - Example: using an array of 8 disks (or a factor of 8 or a multiple of 8)
  - Block-level stripping
    - Splitting blocks of a file across multiple disk drives
- Benefits
  - Increase the throughput of multiple small accesses (page accesses)
  - Reduce the response time of large accesses

RAID Structure

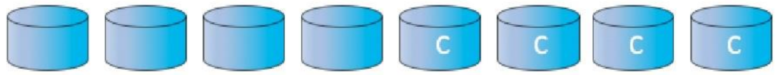
- **RAID** – **redundant array of inexpensive disks**
  - multiple disk drives provides reliability via **redundancy**
- Increases the **mean time to data loss**
- Frequently combined with **NVRAM** to improve write performance
- Several improvements in disk-use techniques involve the use of multiple disks working cooperatively

RAID Levels

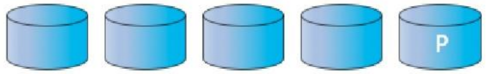
- Disk **striping** uses a group of disks as one storage unit
- RAID is arranged into six different levels
- RAID schemes improve performance and improve the reliability of the storage system by storing redundant data
  - **Mirroring** or **shadowing** (**RAID 1**) keeps duplicate of each disk
  - Striped mirrors (**RAID 1+0**) or mirrored stripes (**RAID 0+1**) provides high performance and high reliability
  - **Block interleaved parity** (**RAID 4, 5, 6**) uses much less redundancy
- RAID within a storage array can still fail if the array fails, so automatic **replication** of the data between arrays is common
- Frequently, a small number of **hot-spare** disks are left unallocated, automatically replacing a failed disk and having data rebuilt onto them



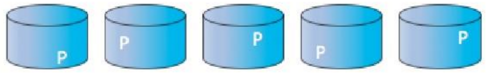
(a) RAID 0: non-redundant striping.



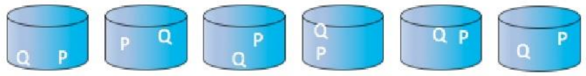
(b) RAID 1: mirrored disks.



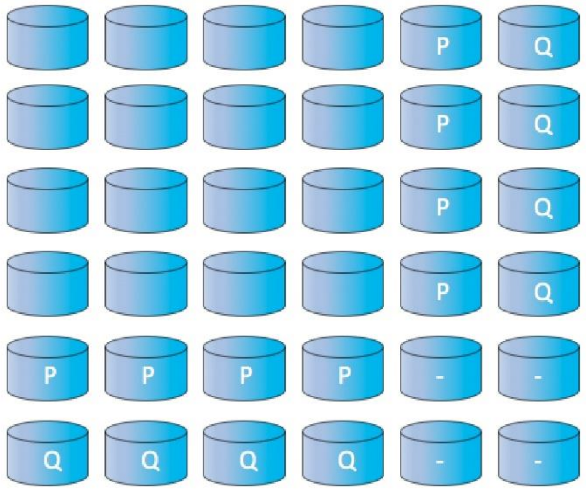
(c) RAID 4: block-interleaved parity.



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



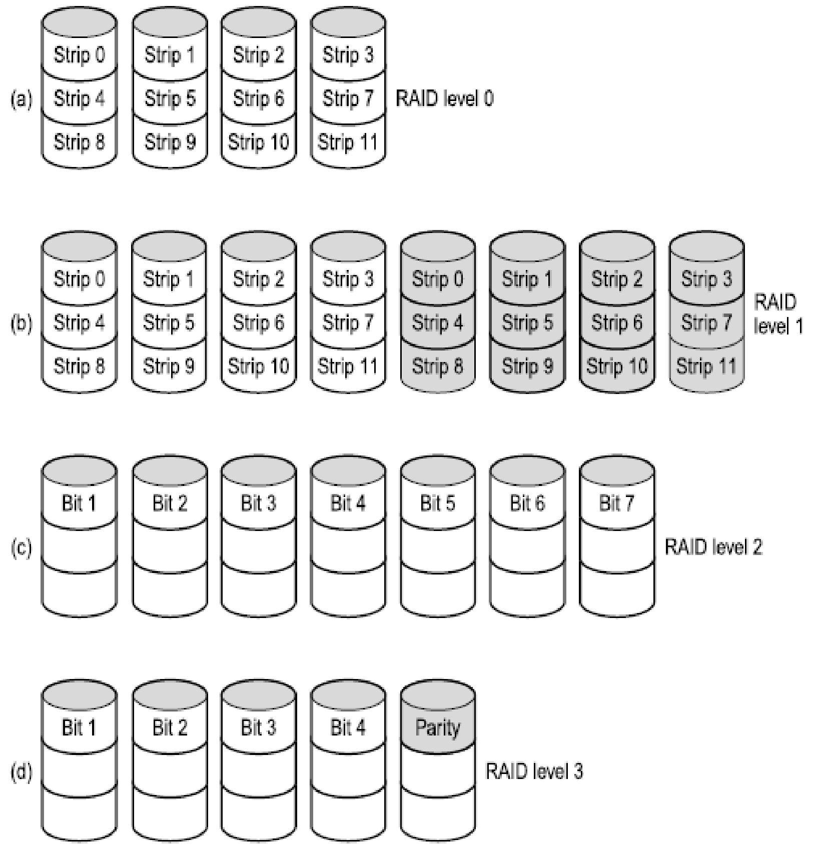
(e) RAID 6: P + Q redundancy.



(f) Multidimensional RAID 6.

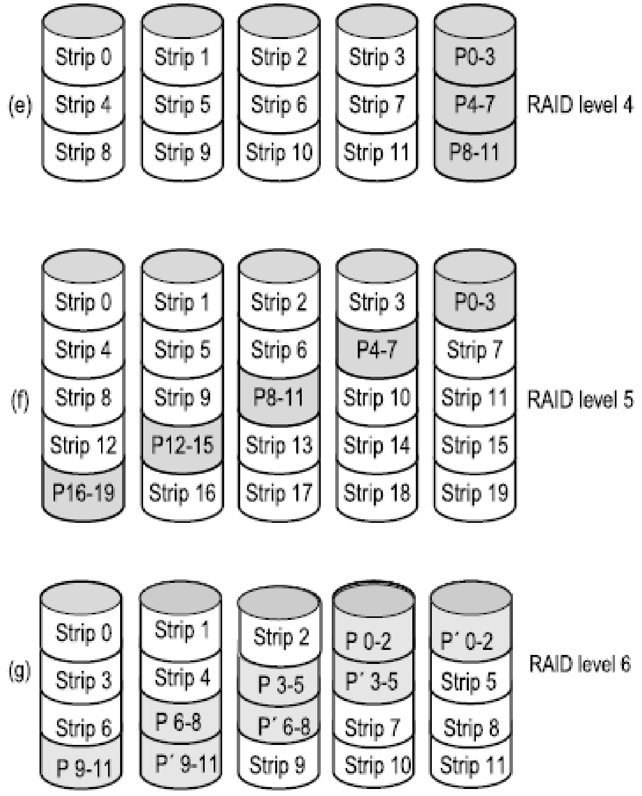
RAID Level 0 - 3

- Level 0 (not a true RAID): Stripping only
- Level 1: Mirror + stripping
- Level 2: Memory-style error-correction-code (ECC) organization
  - Example
    - Split a byte into 4-bit nibbles
    - Add Hamming code (3 bits) to each
    - One bit per drive onto 7 disk drives
- Level 3: Bit-interleaved parity organization
  - Compute a parity bit
  - Driver detects error, a parity bit is sufficient to correct error (unlike memory)
- Backup and parity drives are shown shaded.



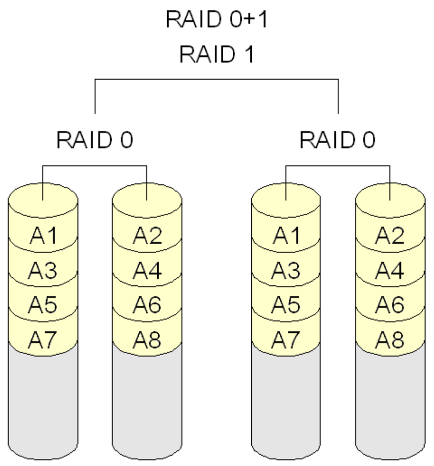
RAID Level 4 - 6

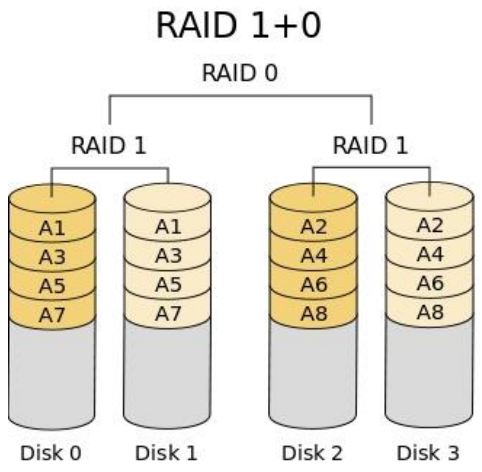
- Level 4: block-interleaved parity organization
  - Stripping
  - Compute parity for blocks
  - One disk for parities
- Level 5: block-interleaved distributed parity
  - Stripping
  - Compute parity for blocks
  - Parities are distributed
- Level 6: P+Q redundancy scheme
  - Extra redundant information to guard multiple disk failures
- Backup and parity drives are shown shaded.



RAID 0+1 and 1+0

- Combination of RAID levels 0 and 1
- RAID 0+1: stripping and then mirroring
- RAID 1+0: mirroring and then stripping





Selecting RAID Level

- RAID 0: high performance, data loss is not critical
- RAID 1: high reliability with fast recovery
- RAID 0+1 & 1+0: both performance and reliability
  - Expense: 2 for 1
- RAID 5:
  - Often preferred for large volumes of data
- Required number of disks?

Other Features

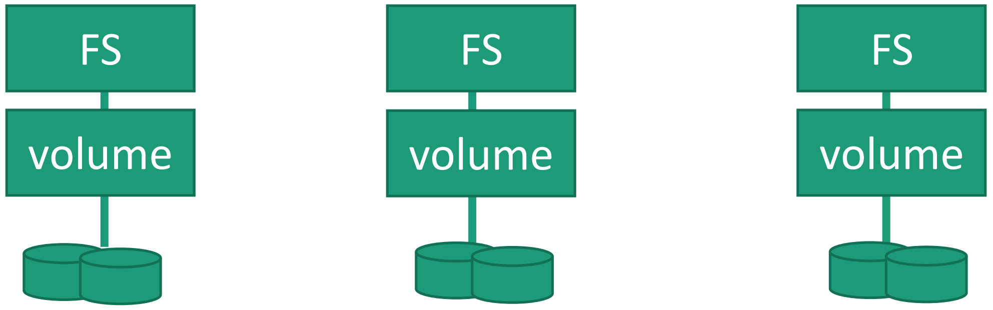
- Regardless of where RAID implemented, other useful features can be added
- Snapshot is a view of file system before a set of changes take place (i.e. at a point in time)
  - More in Ch 12
- Replication is automatic duplication of writes between separate sites
  - For redundancy and disaster recovery
  - 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

Questions

- Reliability and performance
- Performance via parallelism
- Reliability via redundancy
- RAID
  - Which level to use? How many disks?

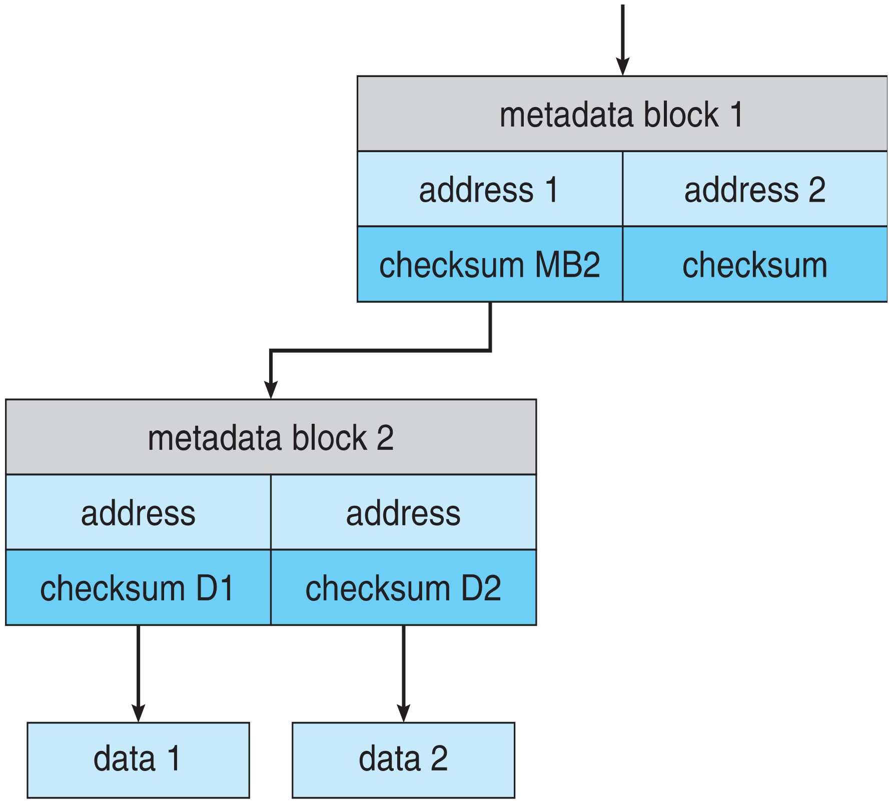
Limitation of RAID

- RAID alone does not prevent or detect data corruption or other errors, just disk failures
- RAID is not flexible
  - Present a disk array as a volume
  - What if a file system is small, or large, or change over time?



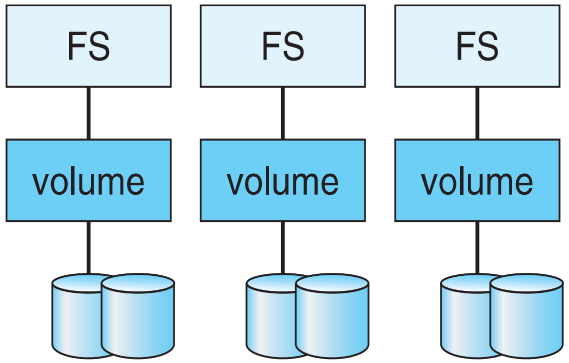
Extensions: Solaris ZFS

- 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

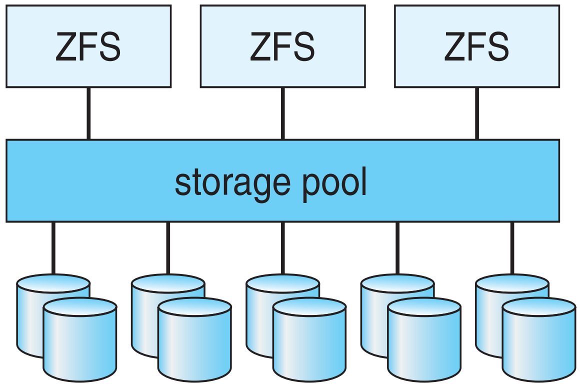


ZFS checksums all metadata and data

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 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

Questions?

- Limitation of RAID?
- ZFS?
- Object storage