Operating System: Chap12 Mass Storage System

National Tsing Hua University

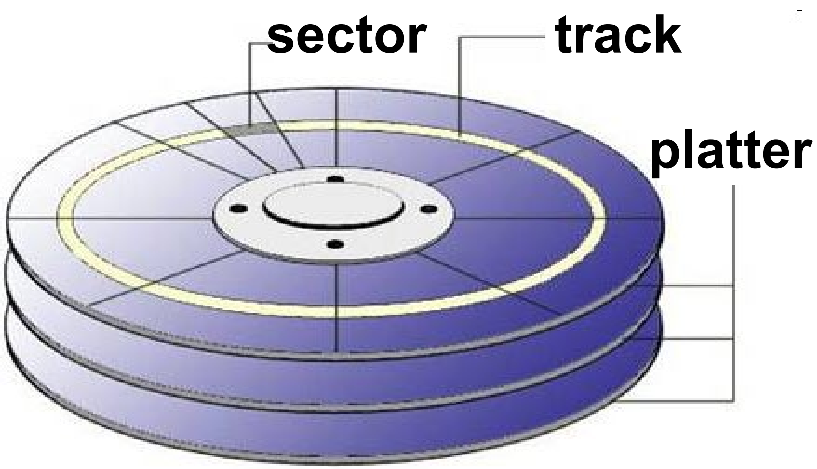
2022, Fall Semester

Overview

- Disk Structure
- Disk Scheduling
- Disk & Swap-Space Management
- RAID

Disk Structure

- Disk drives are addressed as large 1-dim arrays of logical blocks
  - logical block: smallest unit of transfer (**sector**)
- Logical blocks are mapped onto disk sequentially
  - Sector 0: 1st sector of 1st track on the outermost cyl.
  - go from outermost cylinder to innermost one

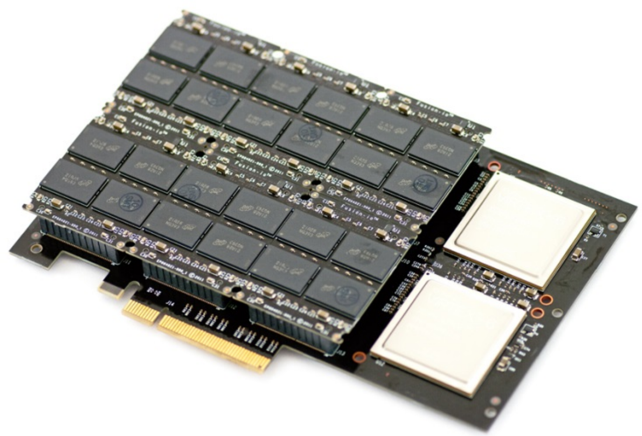


Sectors per Track

- **Constant linear velocity** (CLV)
  - density of bits per track is uniform
  - more sectors on a track in outer cylinders
  - keeping same data rate
    - ➔ increase rotation speed in inner cylinders
  - applications: CD-ROM and DVD-ROM
- **Constant angular velocity** (CAV)
  - keep same rotation speed
  - larger bit density on inner tracks
  - keep same data rate
  - applications: hard disks

Disk IO

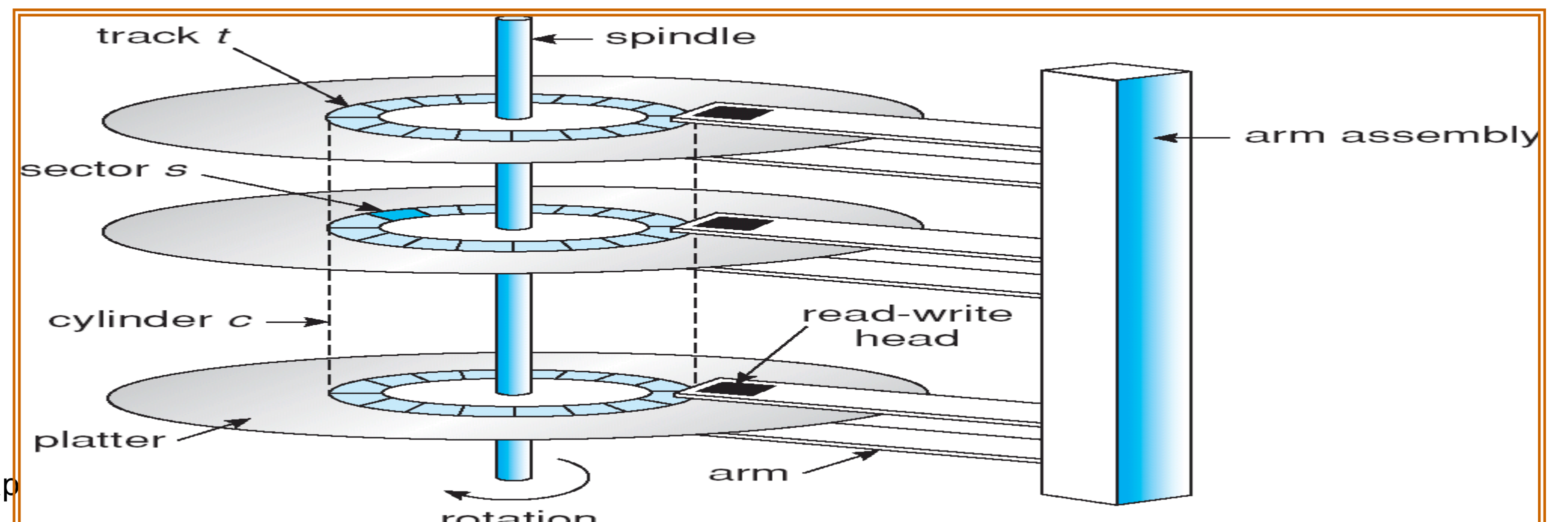
- Disk drive attached to a computer by an I/O bus
  - EIDE, ATA, SATA (Serial ATA), USB, SCSI, etc
  - I/O bus is controlled by controller
    - **Host** controller (computer end)
    - **Disk** controller (built into disk drive)



Disk Scheduling

Introduction

- Disk-access time has 3 major components
  - **seek time**: move disk arm to the desired **cylinder**
  - **rotational latency**: rotate a disk to bring the desired **sector** under the read-write head
  - **read time**: content **transfer time**
- Disk bandwidth:
  - # of bytes transferred/(complete of last req – start of first req)



Disk Scheduling

- Minimize seek time
  - Seek time ≈ seek distance
- Several algorithms exist to schedule the servicing of disk I/O requests
  - **FCFS** (first-come, first-served)
  - **SSTF** (shortest-seek-time-first)
  - **SCAN**
  - **C-SCAN** (circular SCAN)
  - **LOOK and C-LOOK**

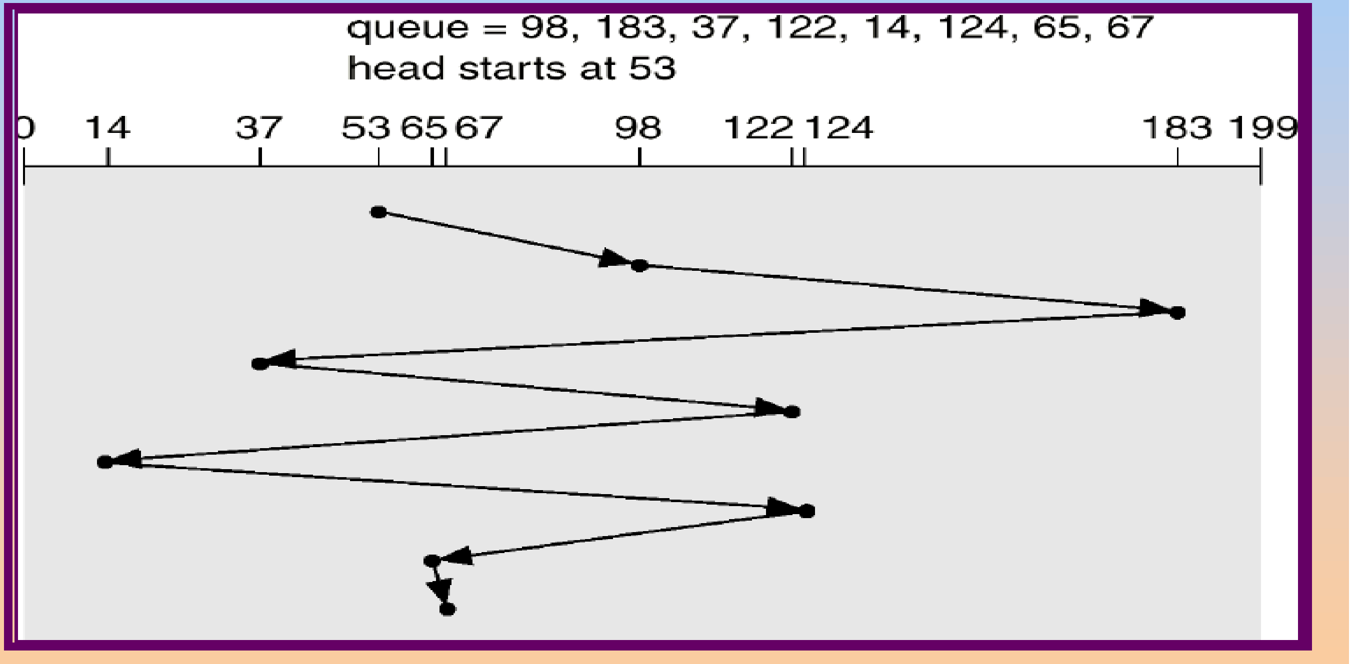
FCFS (First-Come-Frist-Served)

- We illustrate them with a request queue (0-199)

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

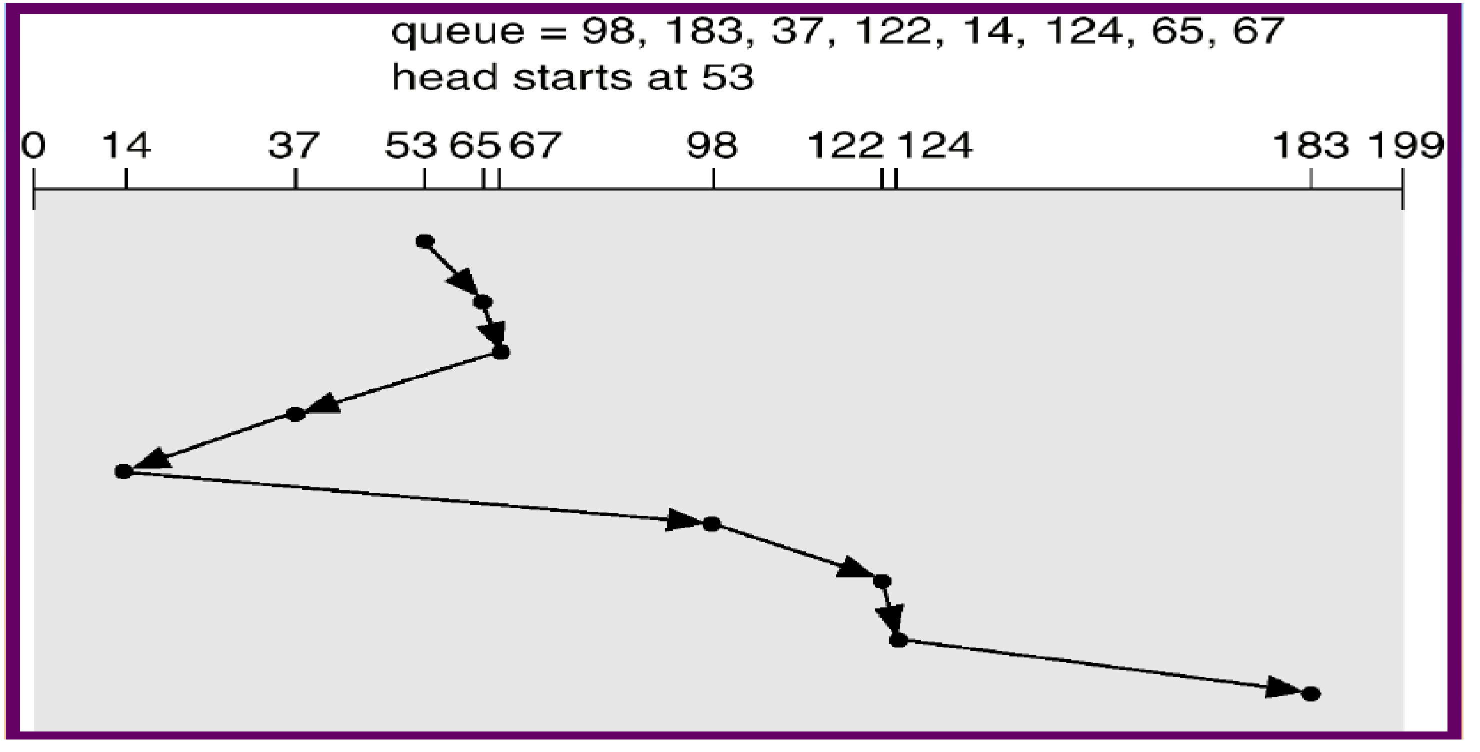
  Head pointer 53

Illustration shows total head movement of 640 cylinders.



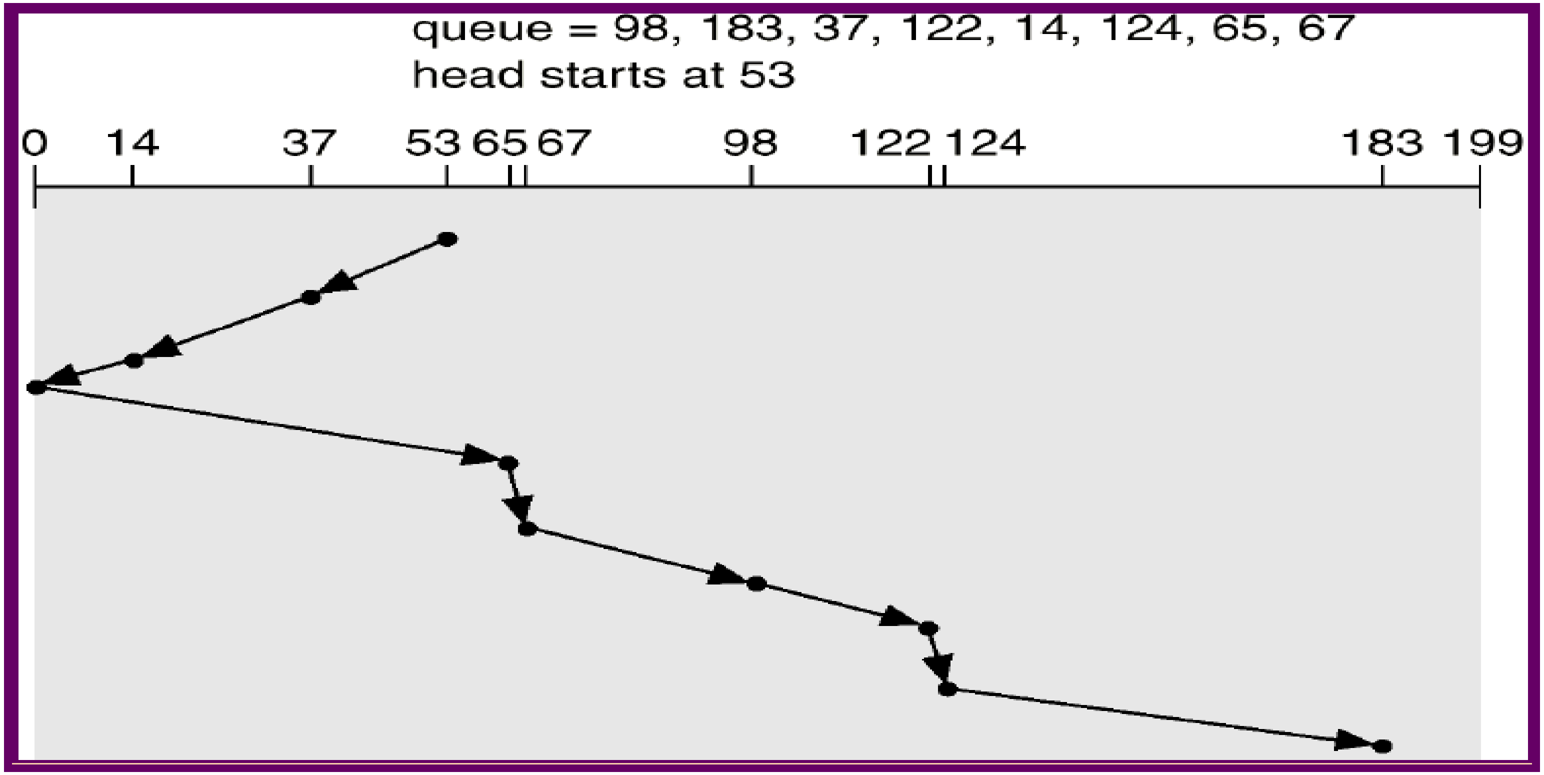
SSTF (Shortest-Seek-Time-First)

- SSTF scheduling is a form of SJF scheduling; may cause starvation of some requests
- total head movement: 236 cylinders



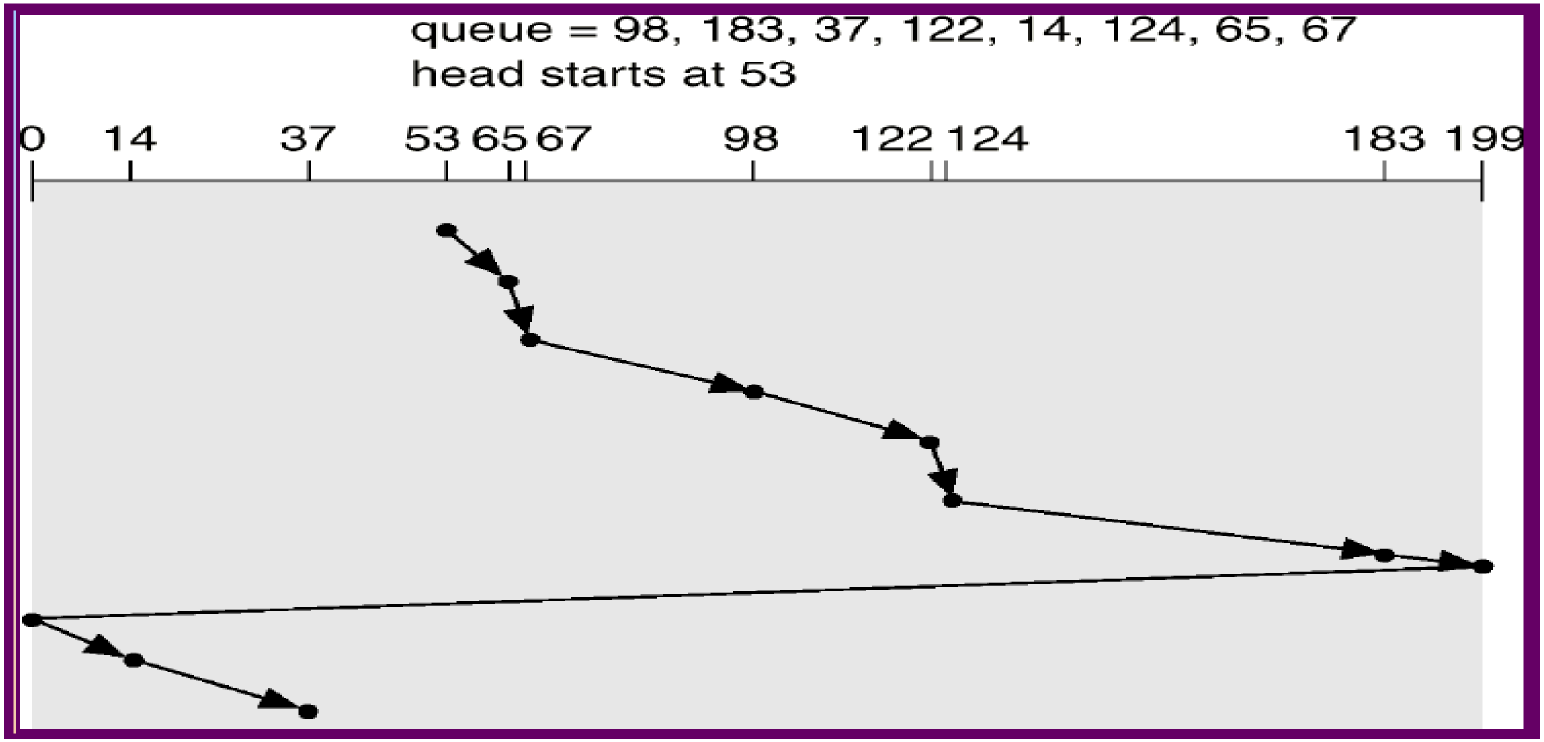
SCAN Scheduling

- disk head move from one end to the other end
- A.k.a. elevator algorithm
- total head movement: 236 cylinders



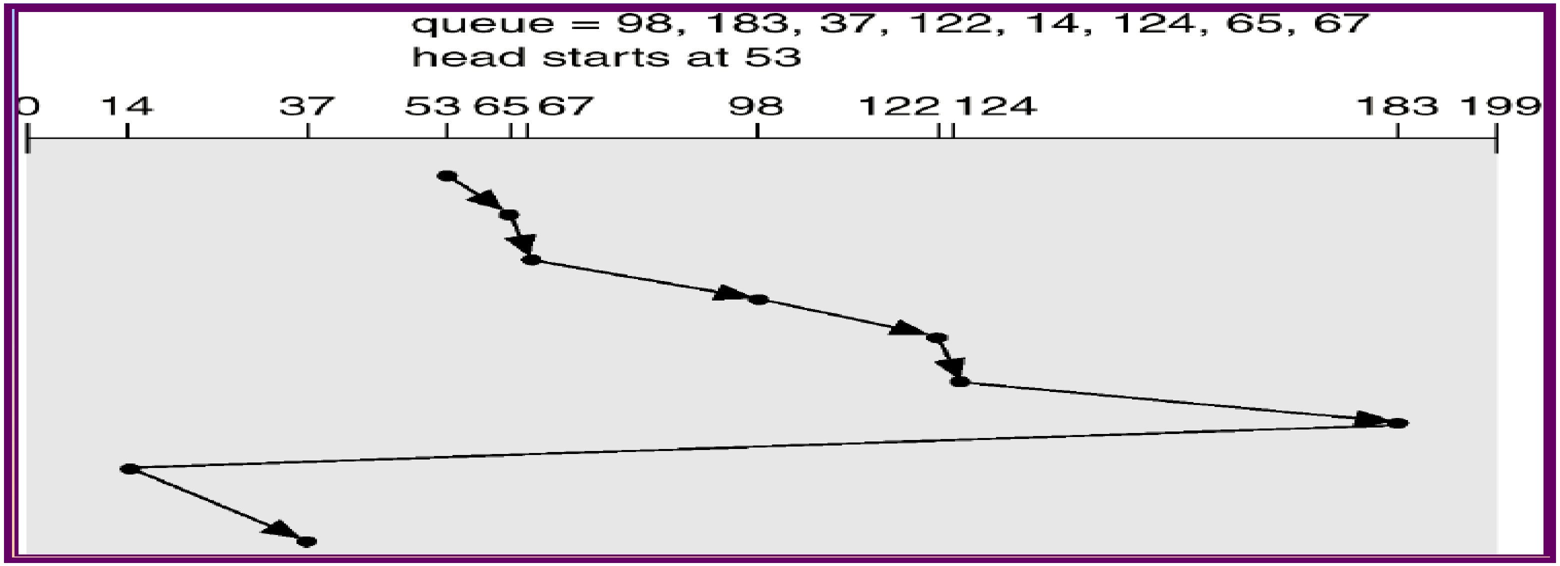
C-SCAN Scheduling

- Disk head move in one direction only
- A variant of SCAN to provide more uniform wait time



C-LOOK Scheduling

- "Looks" ahead to see if there are any requests pending in the direction of head movement
- Disk head moves only to the last request location ☺
- Moving location is affected by requests like SSTF, so it behaves more like SSTF without starvation problem



Selecting Disk-Scheduling Algorithm

- SSTF
  - common and has a natural appeal, but not optimal
- SCAN
  - perform better for disks with heavy load
  - **No** starvation problem
- C-SCAN
  - More uniform wait time
- Performance is also influenced by the file-allocation method
  - Contiguous: less head movement
  - Indexed & linked: greater head movement

Review Slides (I)

- 3 major components in disk-access time
  - Seek
  - Rotation
  - Read
- Goal of disk-scheduling algorithm?
- Disk-scheduling algorithms
  - FCFS
  - SSTF
  - SCAN
  - C-SCAN
  - C-LOOK

Disk Management

- Formatting
- Booting
- Bad block
- Swap space

Disk Formatting

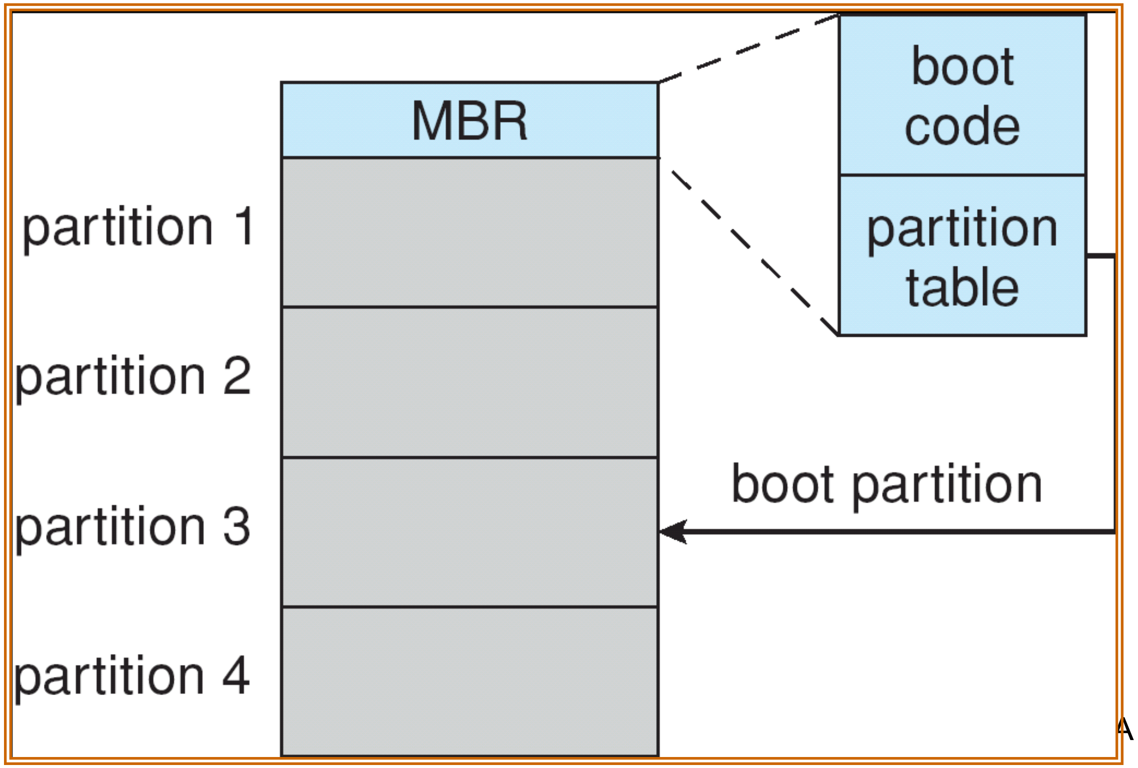
- Low-level formatting (or physical formatting): dividing a disk (magnetic recording material) into sectors that disk controller can read and write
  - each sector = header + data area + trailer
    - header & trailer: sector # and ECC (error-correcting code)
    - ECC is calculated based on all bytes in data area
    - data area size: 512B, 1KB, 4KB
- OS does the next 2 steps to use the disk
  - partition the disk into one or more groups of cylinders
  - logical formatting (i.e. creation of a file system)

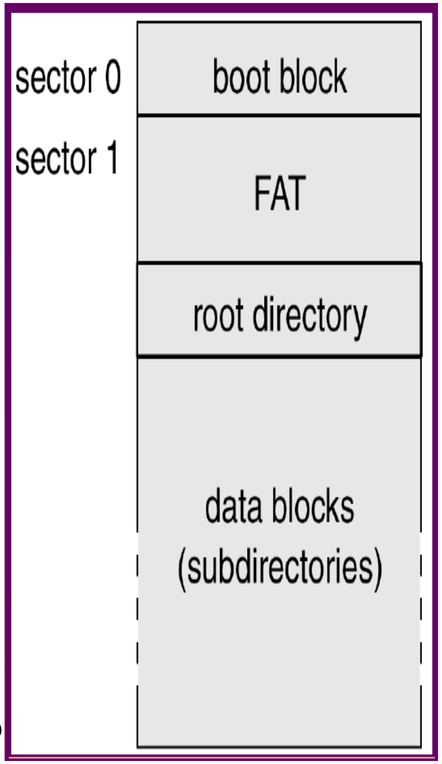
Boot Block

- Bootstrap program
  - Initialize CPU, registers, device controllers, memory, and then starts OS
  - First **bootstrap code** stored in ROM
  - Complete bootstrap in the **boot block** of the boot disk (aka system disk)

Booting from a Disk in Windows 2000

1. Run bootstrap code in ROM
2. Read boot code in MBR(Master boot record)
3. Find boot partition from partition table
4. Read boot sector/block and continue booting





Bad Blocks

- Simple disks like IDE disks
  - Manually use format program to mark the corresponding FAT entry of the bad block
  - Bad blocks are locked away from allocation
- Sophisticated disks like SCSI disks
  - disk controllers maintains the list of bad blocks
  - List is updated over the life of the disk
- **Sector sparing** (forwarding): remap bad block to a spare one
  - Could affect disk-scheduling performance
  - A few spare sectors in each cylinder during formatting
- **Sector slipping**: ships sectors all down one spot

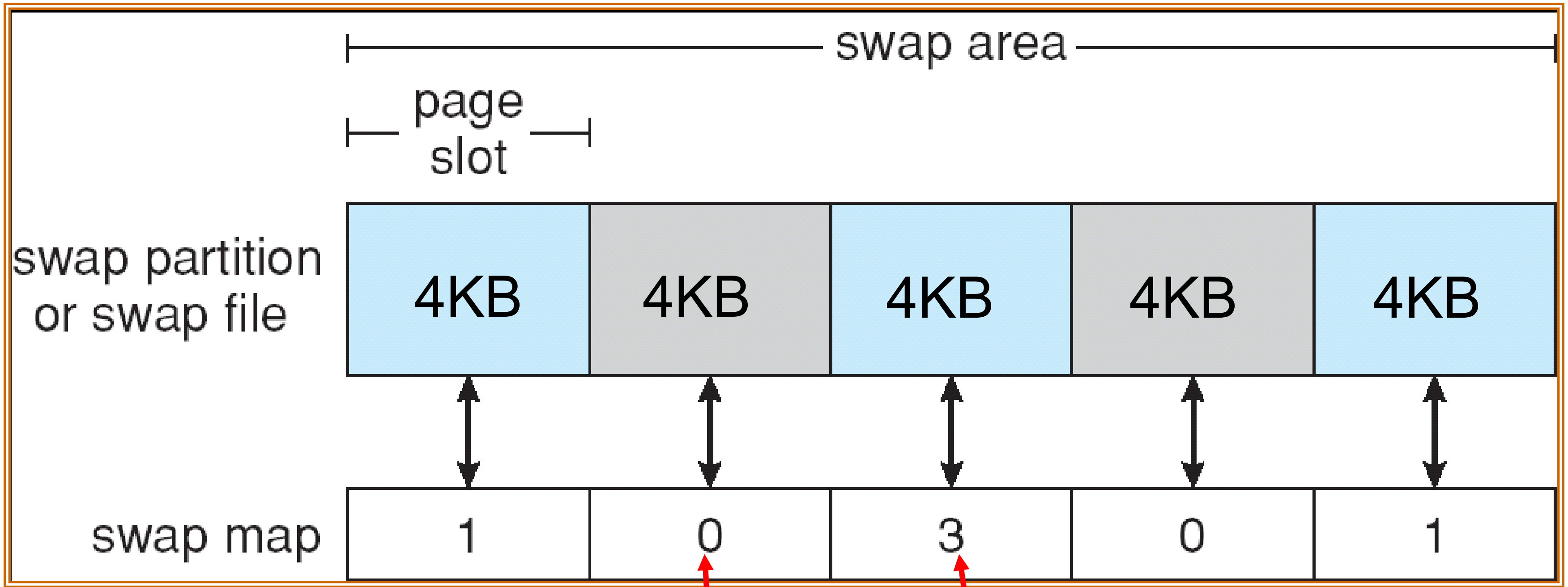
Swap-Space Management

- Swap-space: virtual memory use disk space (swap-space) as an extension of main mem
- UNIX: allows use of multiple swap spaces
- Location
  - part of a normal file system (e.g. NT)
    - Less efficient
  - separate disk partition (raw partition)
    - Size is fixed
  - allows access to both types (e.g. Linux)

Swap Space Allocation

- 1st version: copy entire process between contiguous disk regions and memory
- 2nd version: copy pages to swap space
  - Solaris 1:
    - **text segments** read from file system, thrown away when pageout
    - Only **anonymous memory** (stack, heap, etc) store in swap space
  - Solaris 2:
    - swap-space allocation only when **pageout** rather than **virtual memory creation time**

Data Structures for Swapping (Linux)

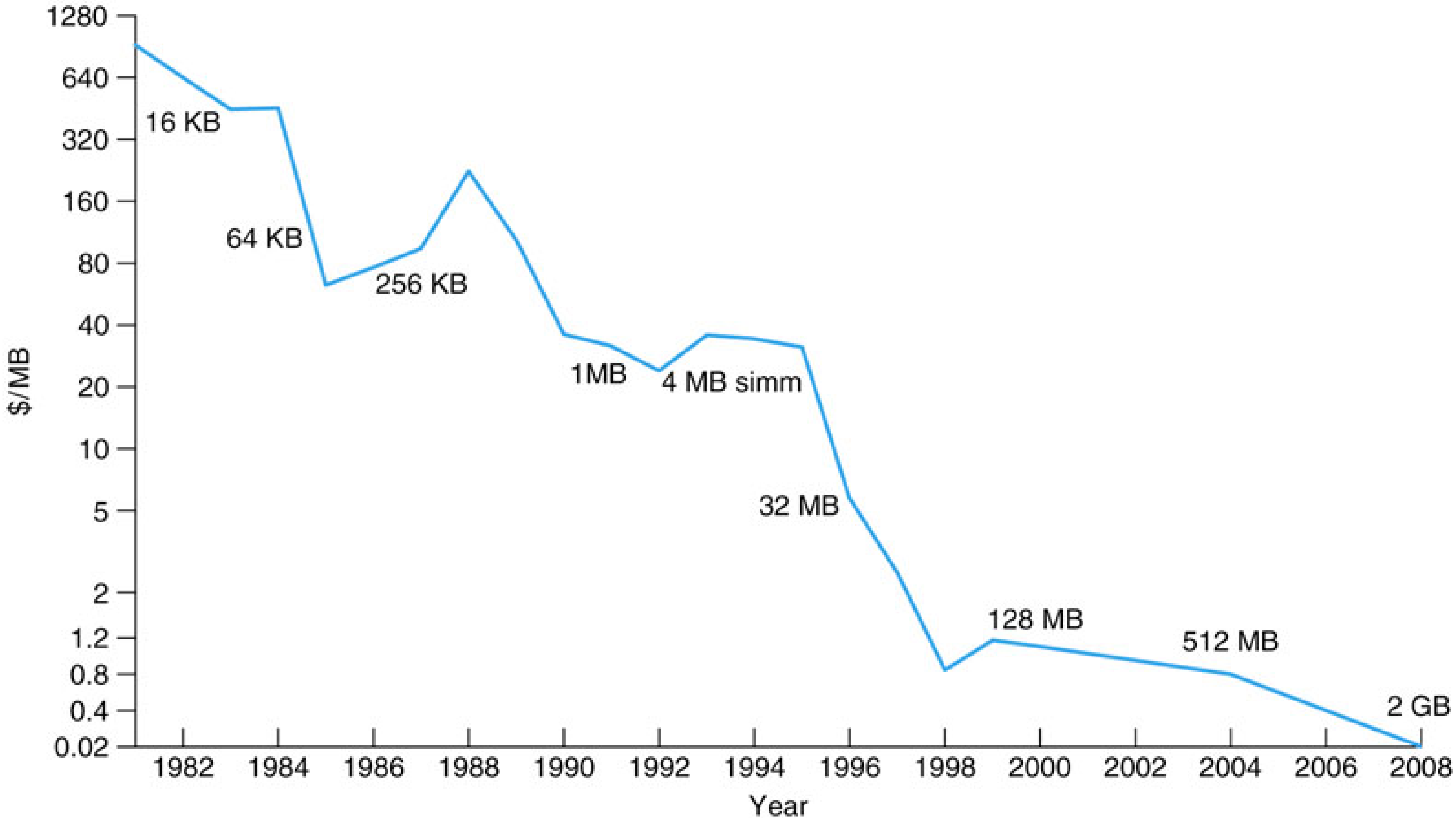


Shared by 3 different processes

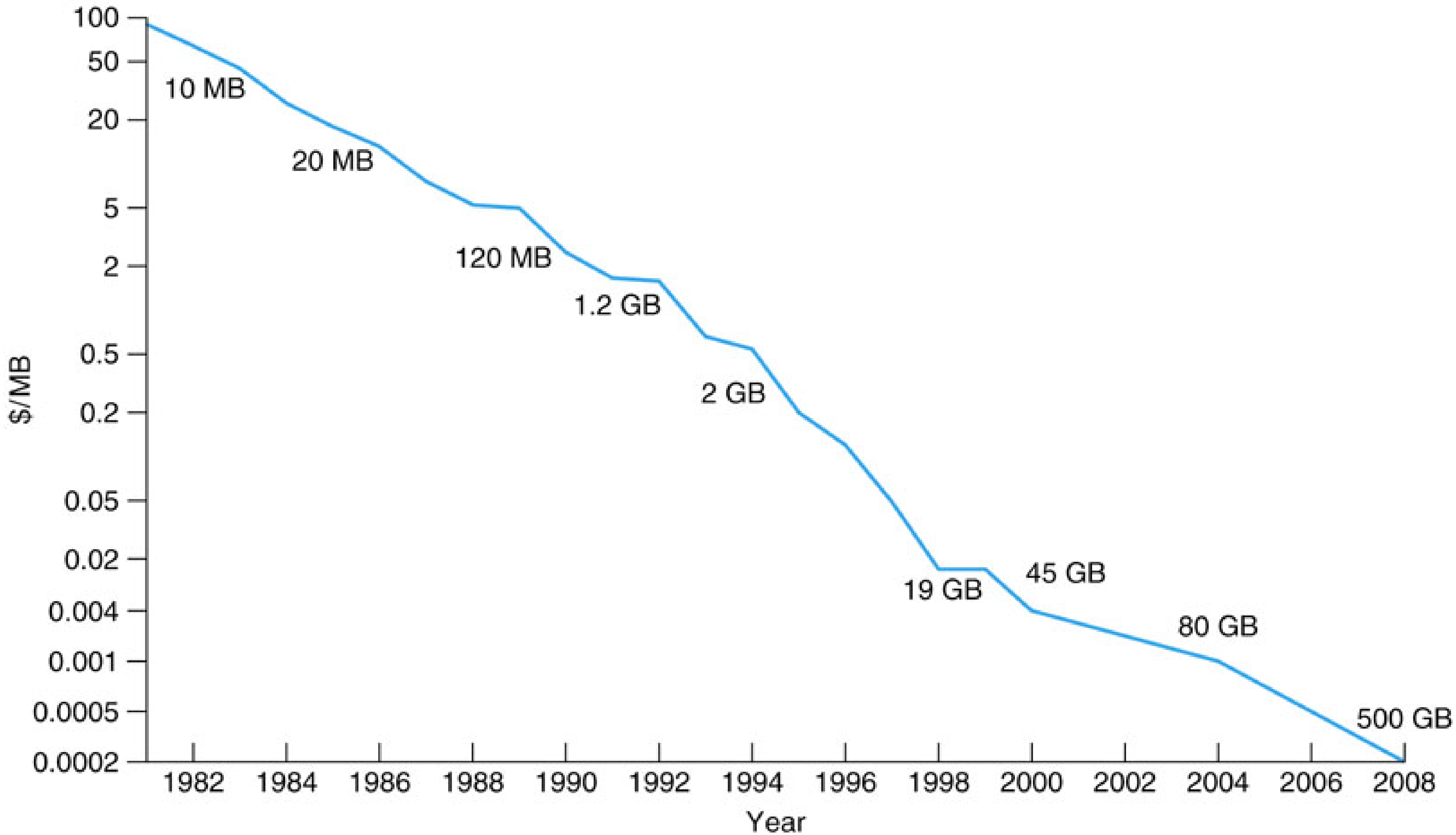
Empty space

RAID Structure

DRAM Price (1981 – 2008)



Magnetic Hard Disk Price (1981 – 2008)

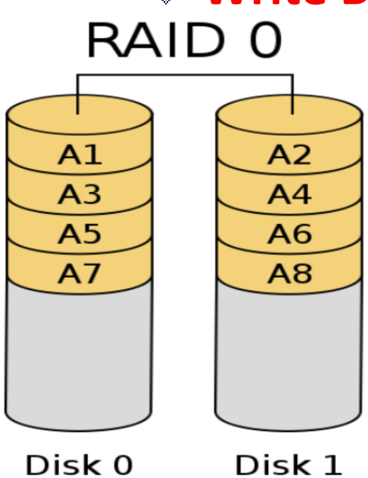


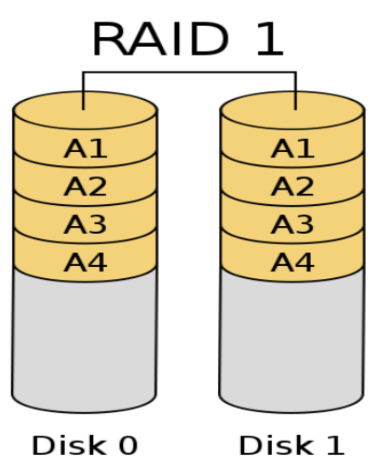
RAID Disks

- **RAID** = **R**edundant **A**rrays of **I**nexpensive **D**isks
  - provide **reliability** via **redundancy**
  - improve **performance** via **parallelism**
- RAID is arranged into different levels
  - Striping
  - Mirror (Replication)
  - Error-correcting code (ECC) & Parity bit

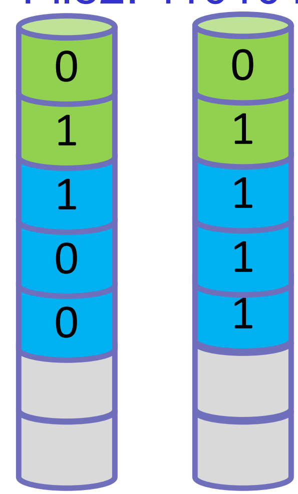
RAID 0 & RAID 1

- RAID 0: non-redundant **striping**
  - Improve **performance** via **parallelism**
  - I/O bandwidth is proportional to the striping count
    - **Both read and write BW increase by N times (N is the number of disks)**
- RAID 1: **Mirrored** disks
  - Provide **reliability** via **redundancy**
    - **Read BW increases by N times**
    - **Write BW remains the same**

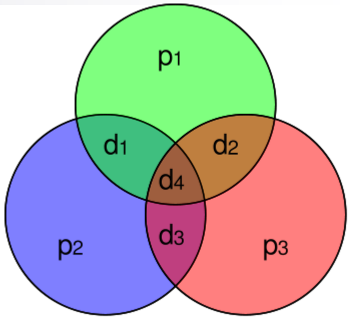


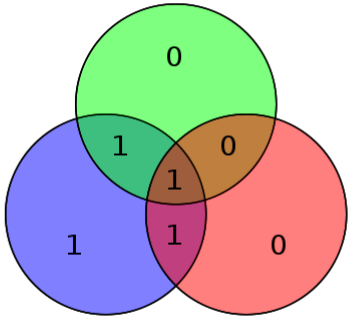


RAID0 Example
File1: 0011
File2: 110101



RAID 2: Hamming code

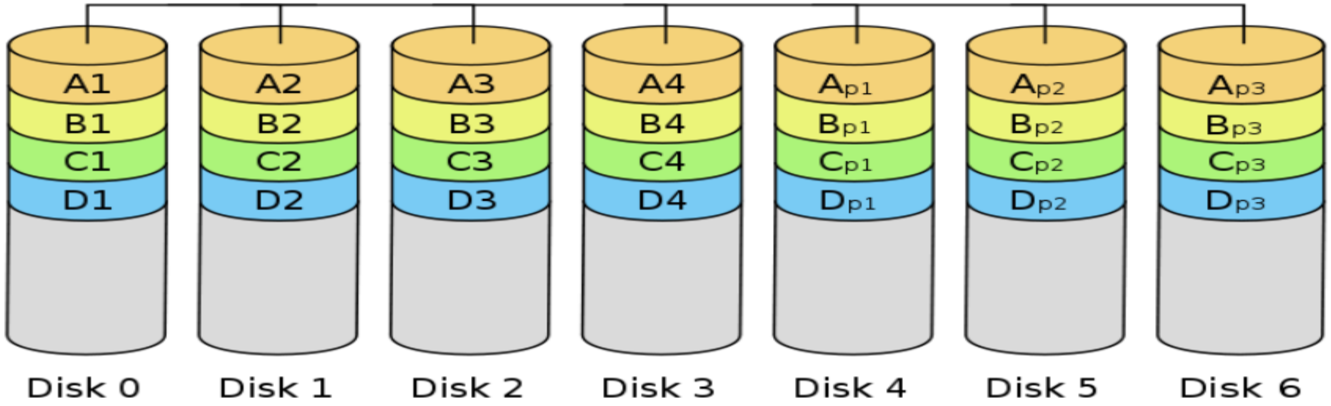




- E.g.: Hamming code(7,4)
  - 4 data bits (on 4 disks) + 3 parity bits (on 3 disks)
  - Each parity bit is linear code of 3 data bits

☺Recover from any single disk failure
  - Can detect up to two disks(i.e. bits) error
  - But can only "correct" one bit error

☺Better space efficient than RAID1 (75% overhead)



Hamming code reference: http://en.wikipedia.org/wiki/Hamming_code

RAID 3 & 4: Parity Bit

- Disk controller can detect whether a sector has been read correctly

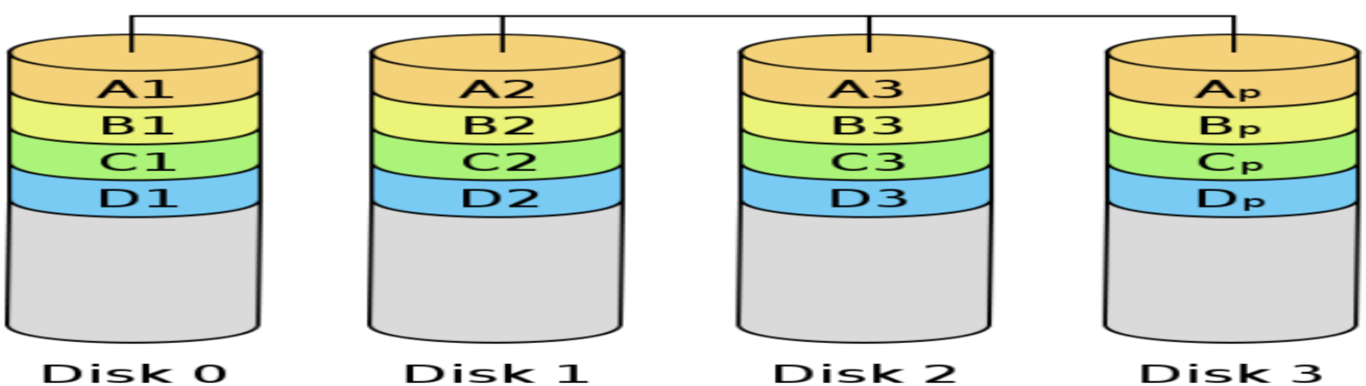
➔ a single parity bit is enough to correct error from a single disk failure

- **RAID 3**: Bit-level striping; **RAID 4**: Block-level striping

☺ Even better space efficiency (33% overhead)

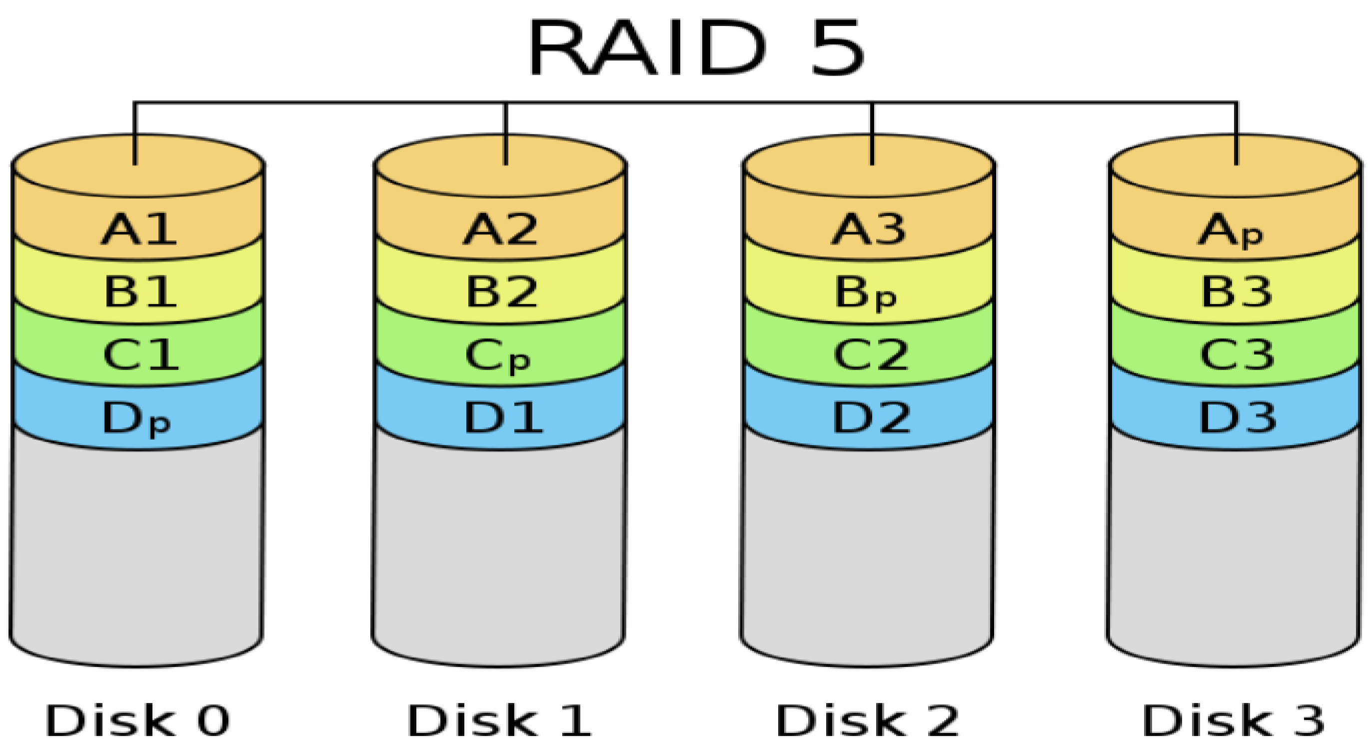
☹ Cost to compute & store parity bit

- RAID4 has higher I/O throughput, because controller does not need to reconstruct block from multiple disks



RAID 5: Distributed Parity

- Spread data & parity across all disks
- Prevent over use of a single disk (e.g. RAID 3,4)

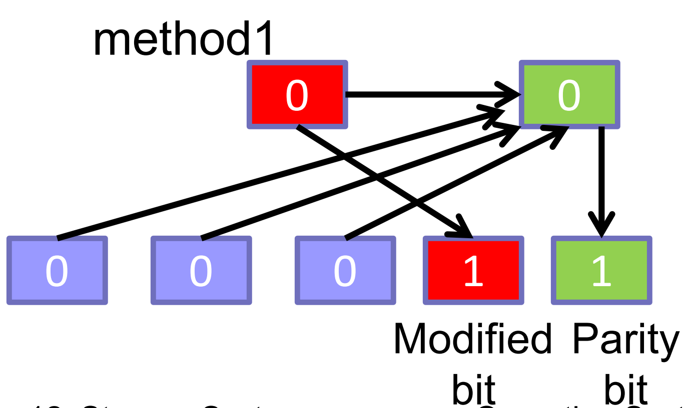


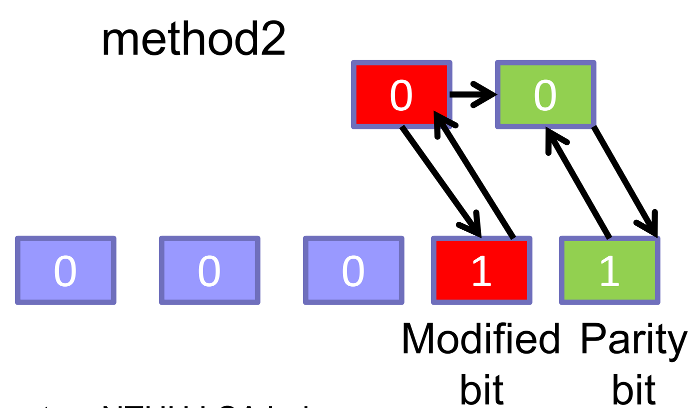
RAID 5: Distributed Parity

- Read BW increases by N times, because all five disks can serve a read request
- Write BW
  - Method1: (1)read out all unmodified (N-2) data bits. (2) re-compute parity bit. (3) write both modified bit and parity bit to disks.

    ➔ write BW = N / ((N-2)+2) = 1 ➔ remains the same
  - Method2: (1)only read the parity bit and modified bit. (2) re-compute parity bit by the difference. (3) write both modified bit and parity bit.

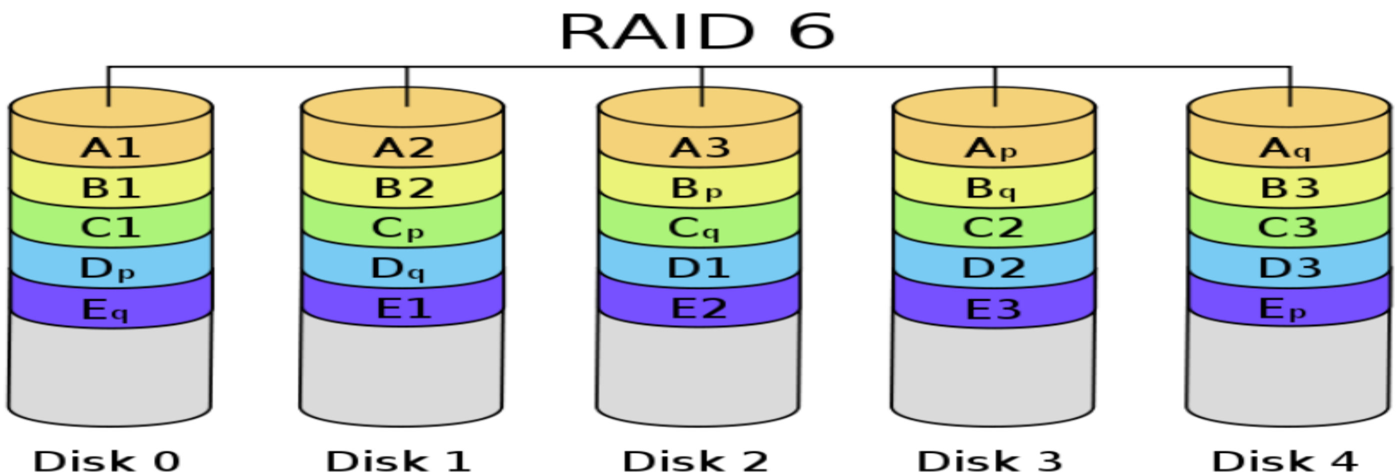
    ➔ write BW = N / (2+2) = N/4 times faster





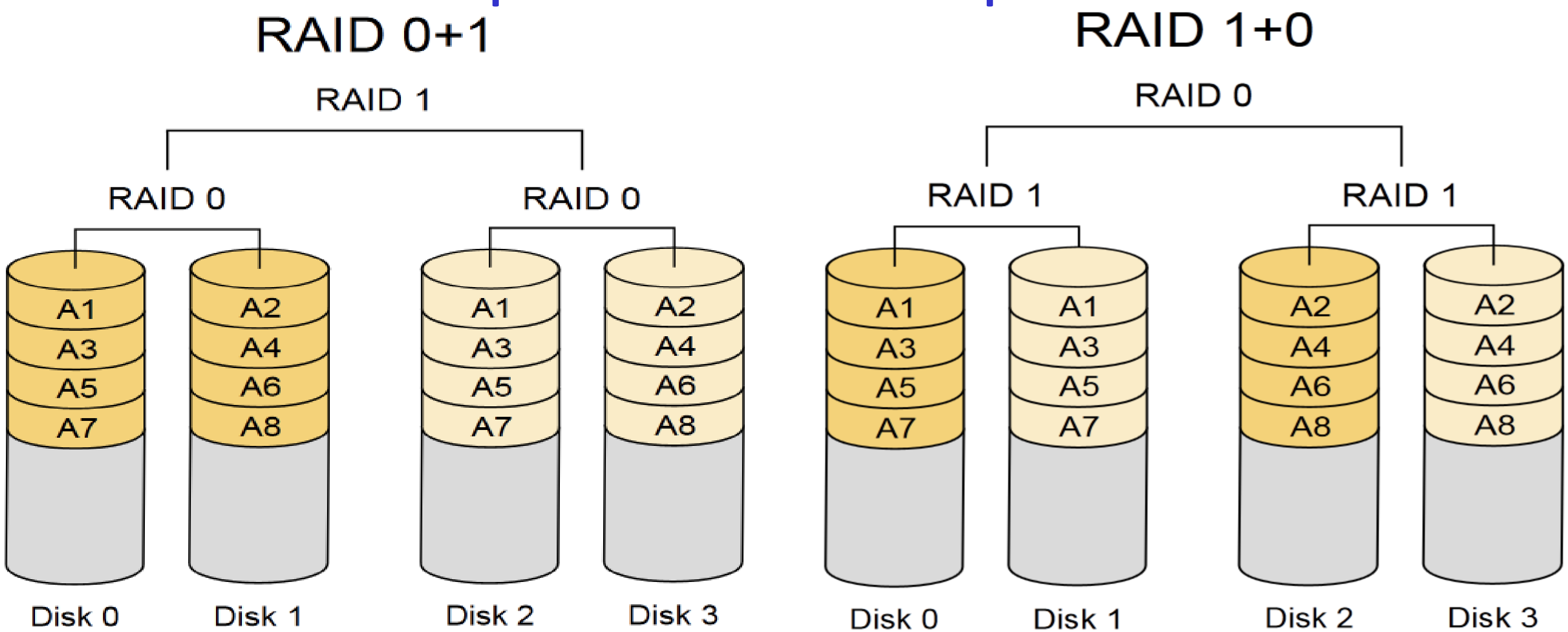
RAID 6: P+Q Dual Parity Redundancy

- Like RAID 5, but stores extra redundant information to guard against **multiple disk failure**
- Use ECE code (i.e. Error Correction Code) instead of single parity bit
- Parity bits are also striped across disks



Hybrid RAID

- RAID 0+1: Stripe then replicate
- RAID 1+0: Replicate then stripe



*First level often control by a controller. Therefore, RAID 10 has better fault tolerance than RAID 01 when multiple disk fails

http://www.thegeekstuff.com/2011/10/raid10-vs-raid01/

Review Slides (II)

- Swap space using FS? Raw partition?
- How to reduce swap space usage?
- RAID disks? Purpose?
- RAID-0~6?
- RAID 0+1, RAID 1+0

Textbook Questions

- 12.1: None of the disk-scheduling disciplines, except FCFS, is truly fair (starvation may occur).
  a. Explain why this assertion is true.
  b. Describe a way to modify algorithms such as SCAN to ensure fairness.
  c. Explain why fairness is an important goal in a time-sharing system.
  d. Give three or more examples of circumstances in which it is important that the operating system be unfair in serving I/O requests
- 12.3: Suppose that a disk drive has 5000 cylinders, numbered 0 to 4999. The drive is currently serving a request at cylinder 2150, and the previous request was at cylinder 1805. The queue of pending requests, in FIFO order, is 2069, 1212, 2296, 2800, 544, 1618, 356, 1523, 4965, 3681. Starting from the current head position, what is the total distance (in cylinders) that the disk arm moves to satisfy all the pending requests, for each of the following disk scheduling algorithms: a. FCFS b. SSTF c. SCAN d. LOOK e. C-SCAN f. C-LOOK

Textbook Questions

- 12.8: Disk requests are not usually uniformly distributed. For example, we can expect a cylinder containing the file-system metadata to be accessed more frequently than a cylinder containing only files. Suppose you know that 50 percent of the requests are for a small, fixed number of cylinders. Would any of the disk scheduling algorithms discussed during the last lecture be particularly good for this case? (i.e. Would any of the disk scheduling algorithms perform particularly better than the other ones for this case?) Explain clearly.

Sector Sparing Example

- OS tries to read block 87
- controller finds out 87 is bad block, reports to OS
- next time system rebooted, controller replaces the bad block with a spare
- OS requests block 87 again, controller reads the spare block instead