國立陽明交通大學
NATIONAL YANG MING CHIAO TUNG UNIVERSITY

Institute of Artificial Intelligence Innovation
Department of Computer Science

Operating System

Lecture 11: Mass Storage System

Shuo-Han Chen 陳碩漢
shch@nycu.edu.tw

Wed. 10:10 - 12:00 EC115 +
Fri. 11:10 – 12:00 Online

Course Schedule

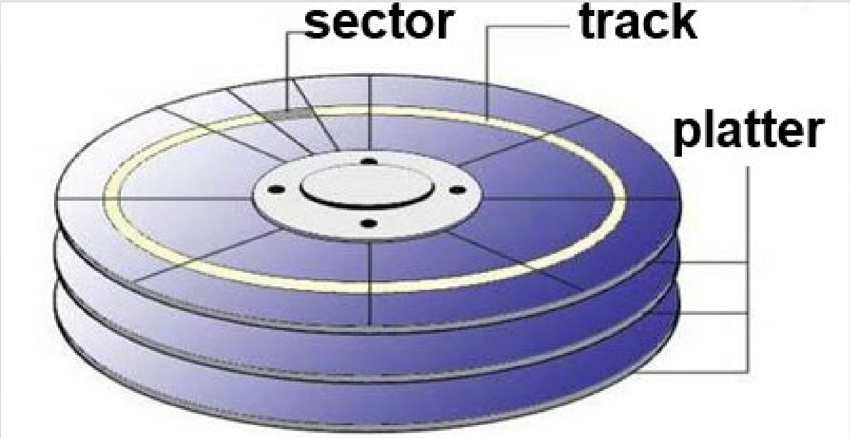
W	Date	Lecture	Online	Homework
1	Sept. 4	Lec00: Couse Overview & Historical Prospective		
2	Sept. 11	Lec01: Introduction	V	
3	Sept. 18	Lec02: OS Structure	V	HW01 Due 10/5
4	Sept. 25	Lec03: Processes Concept	X	
5	Oct. 2	Typhoon – No class	V	
6	Oct. 9	Lec07: Memory Management	V	
7	Oct. 16	Lec08: Virtual Memory Management	V	HW02 Due 11/2
8	Oct. 23	Lec04: Multithreaded Programming	V	
9	Oct. 30	Midterm Exam		
10	Nov. 6	Lec05: Process Scheduling	V	Let's take a breath
11	Nov. 13	Lec06: Process Synchronization & Deadlocks	X	HW03
12	Nov. 20	School Event – No class	V	
13	Nov. 27	Lec09: File System Interface	V	
14	Dec. 4	Lec10: File System Implementation	V	HW04
15	Dec. 11	Lec11: Mass Storage System & Lec12: IO Systems	V	
16	Dec. 18	School Final Exam		

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



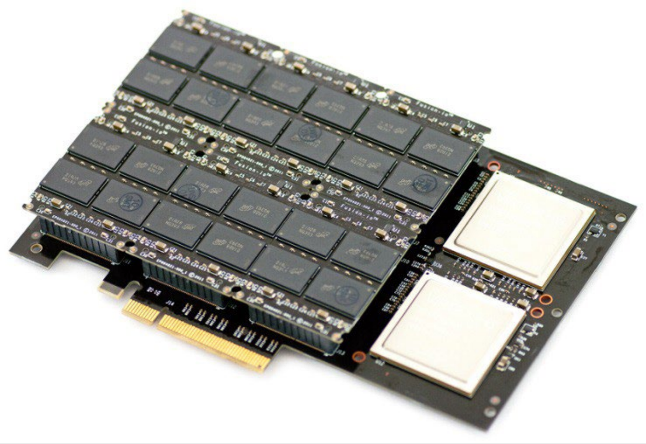
Sectors per Track

- 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

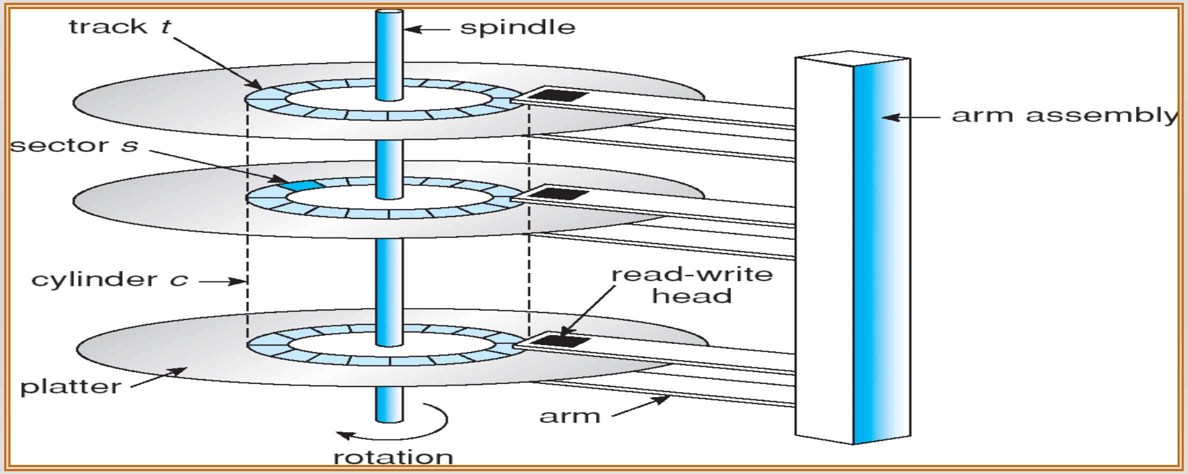
- Disk drive attached to a computer by an I/O bus
  - EIDE, ATA, SATA (Serial ATA), USB, SCSI, etc
  - I/O bus is controlled by controller
    - Host controller (computer end)
    - Disk controller (built into disk drive)


Disk Scheduling

Introduction

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



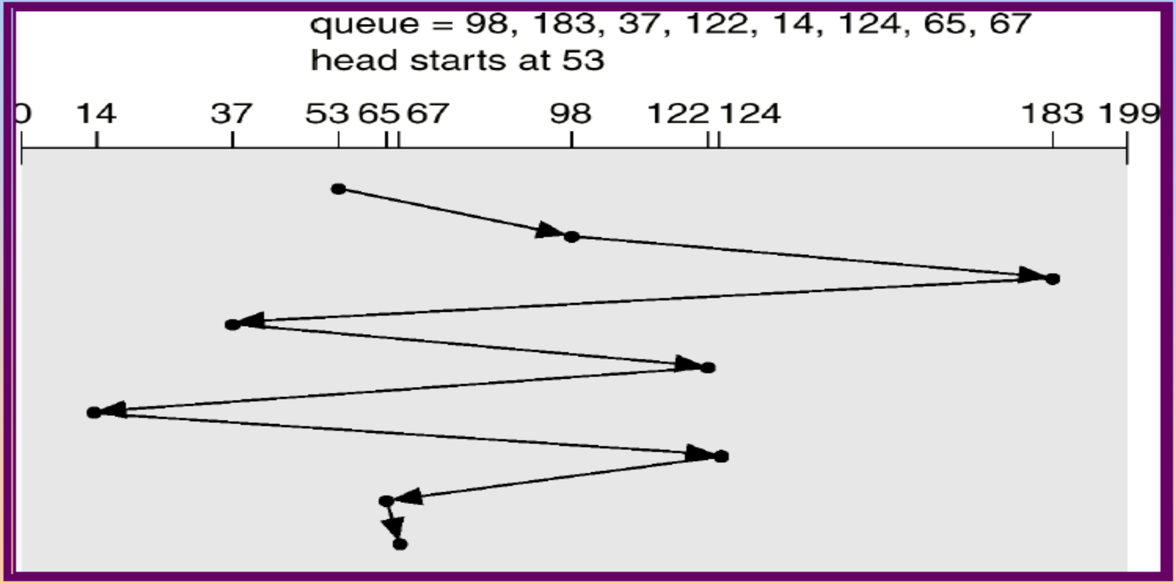
Disk Scheduling

- 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)

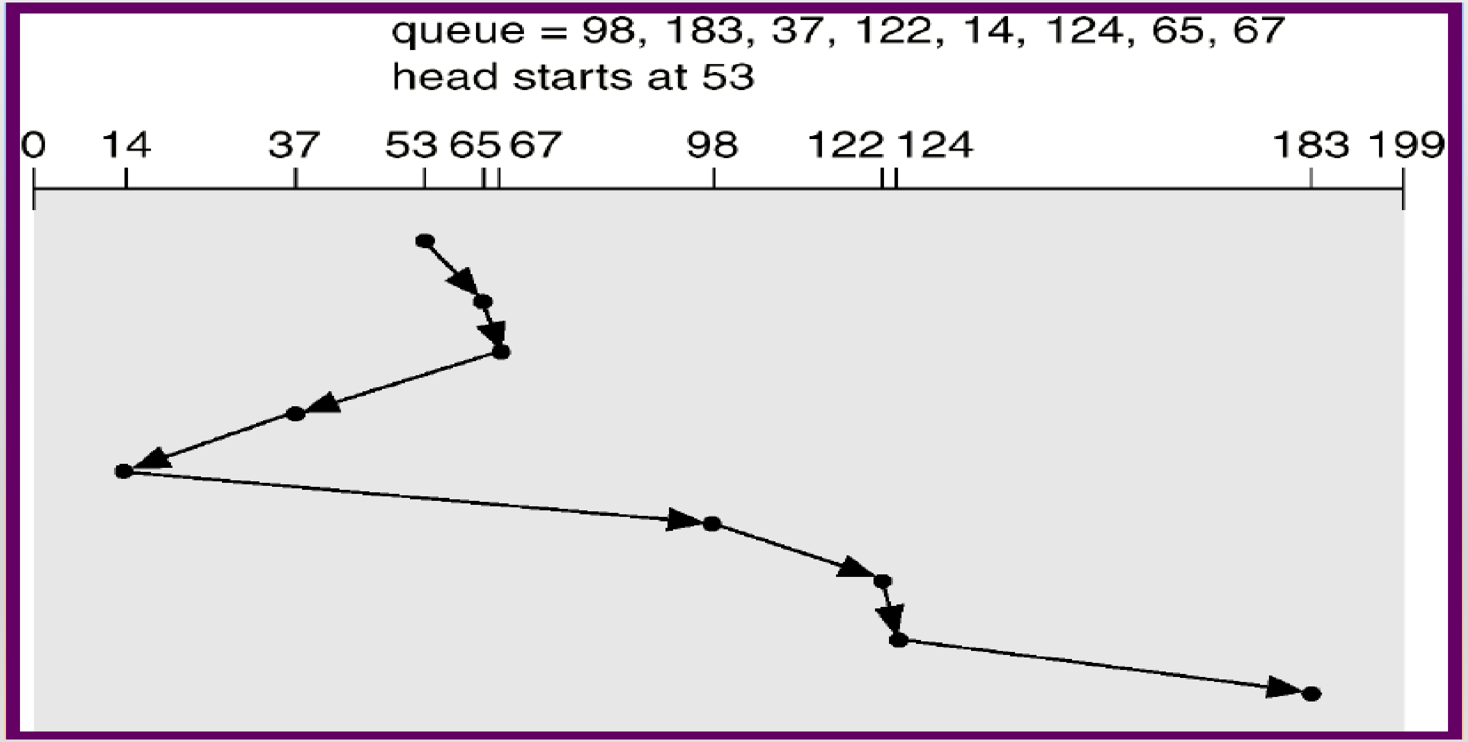
- We illustrate them with a request queue (0-199)
  - 98, 183, 37, 122, 14, 124, 65, 67
- Head pointer 53

Illustration shows total head movement of 640 cylinders.



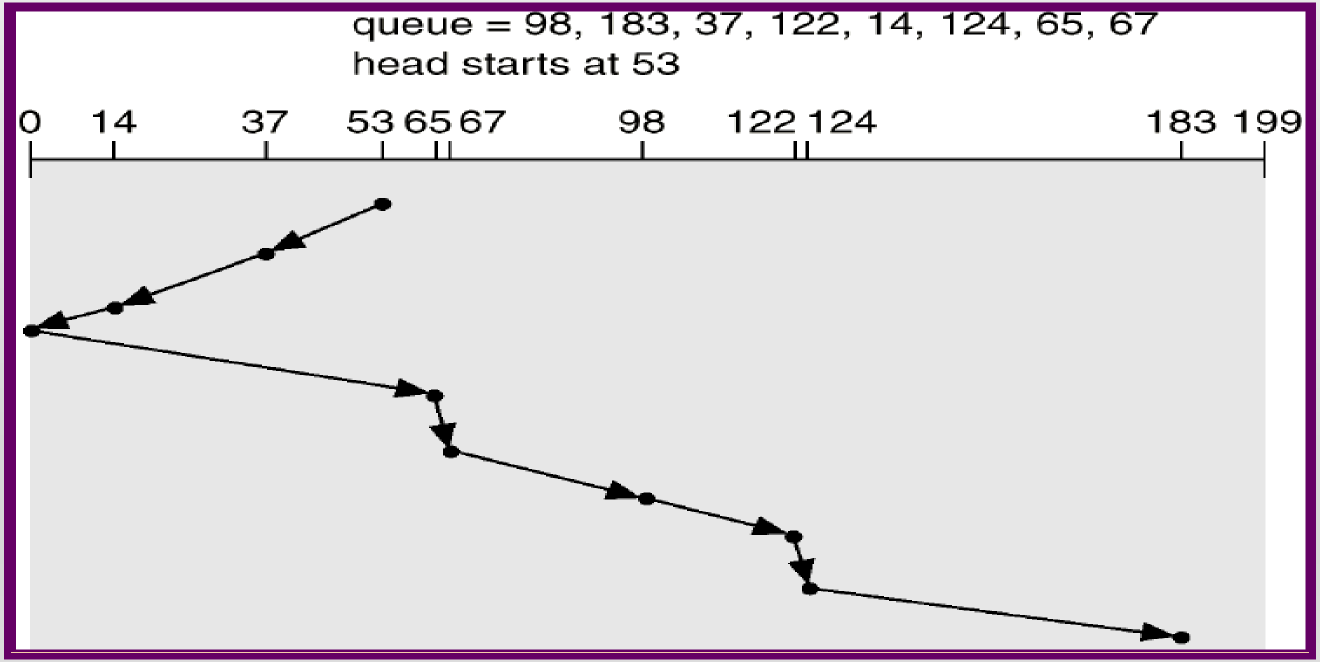
SSTF (Shortest-Seek-Time-First)

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



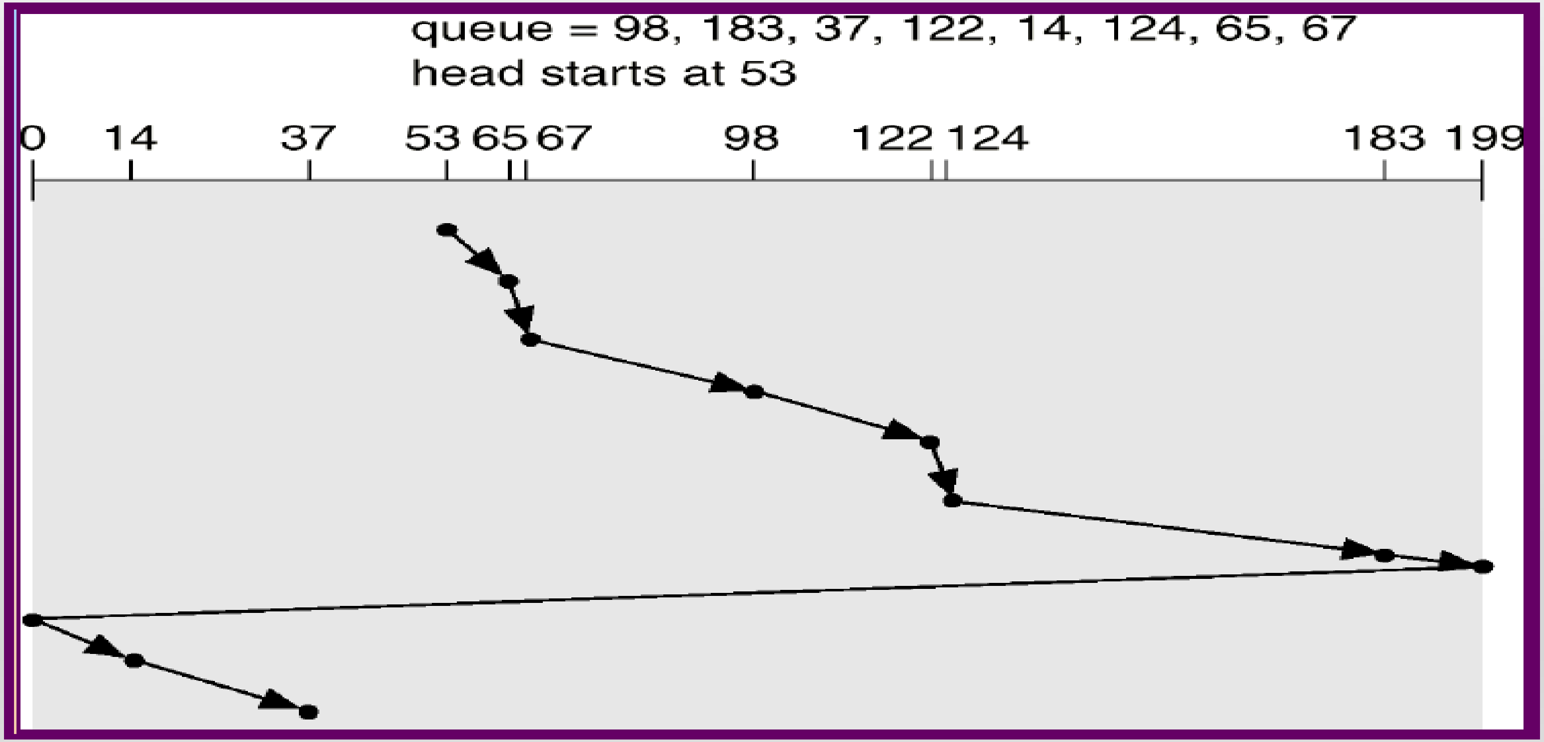
SCAN Scheduling

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



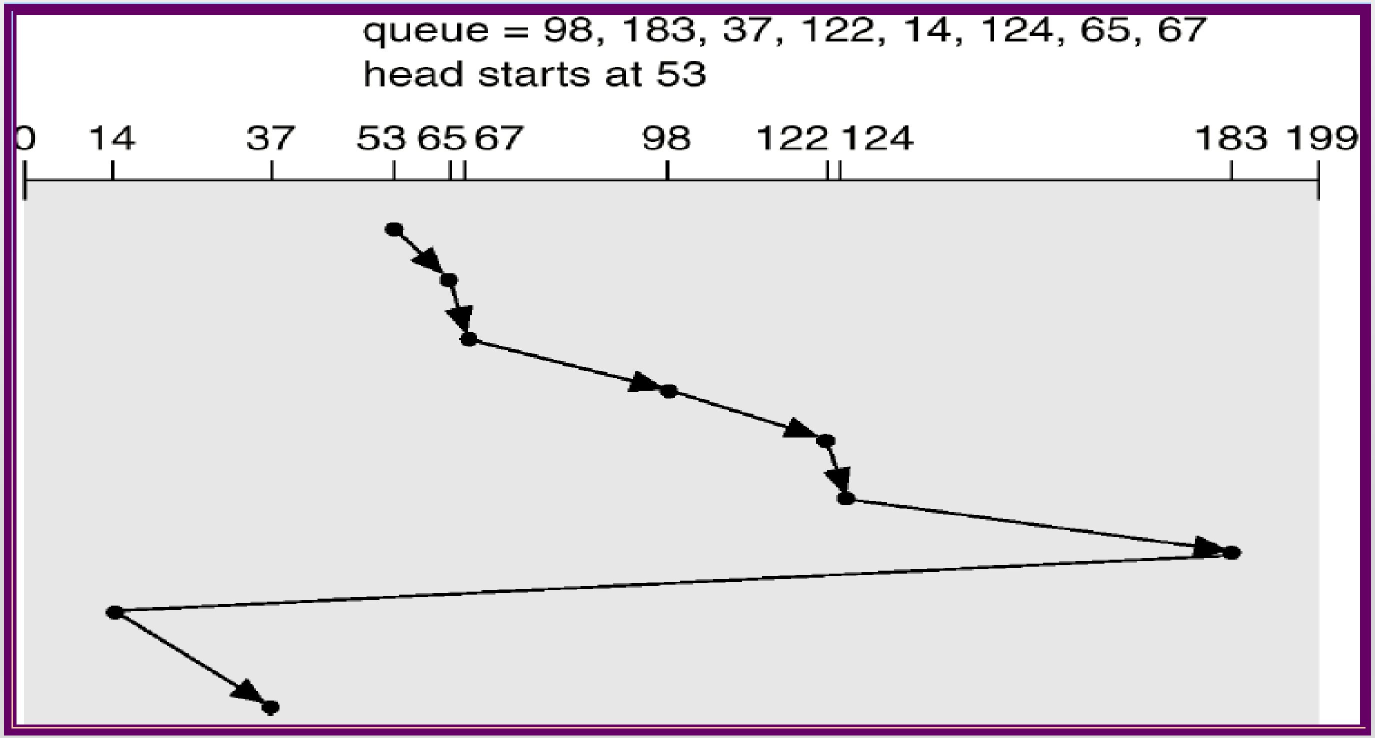
C-SCAN Scheduling

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



C-LOOK Scheduling

- version of C-SCAN
- Disk head moves only to the last request location



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

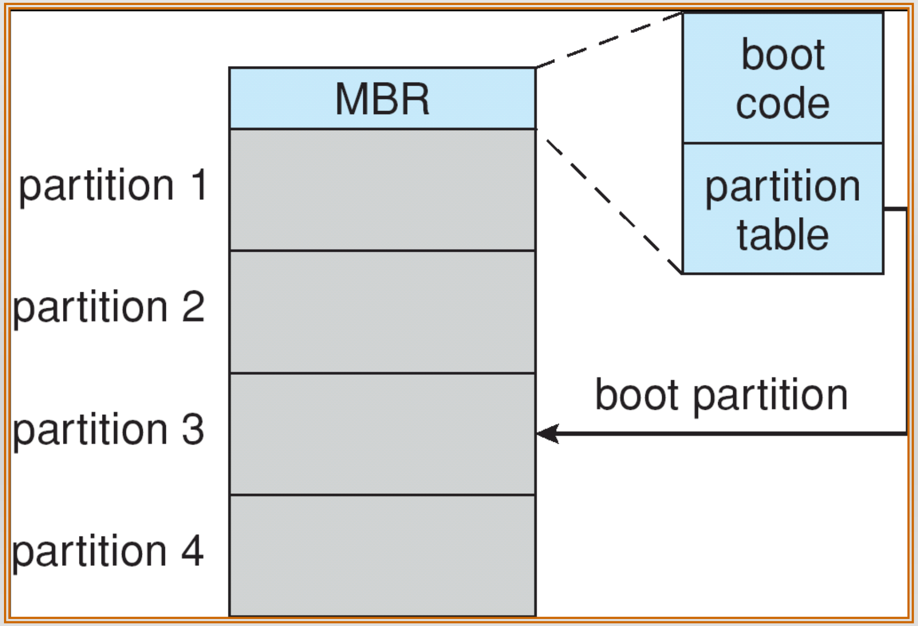
- Low-level formatting (or physical formatting): dividing a disk (magnetic recording material) into sectors that disk controller can read and write
  - each sector = header + data area + trailer
    - header & trailer: sector # and ECC (error-correcting code)
    - ECC is calculated based on all bytes 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)

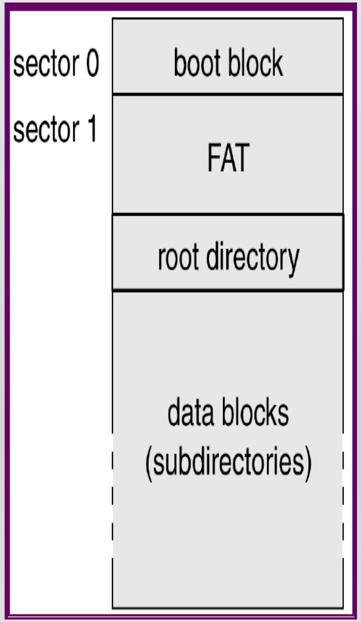
Boot Block

- 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

- Run bootstrap code in ROM
- Read boot code in MBR(Master boot record)
- Find boot partition from partition table
- 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

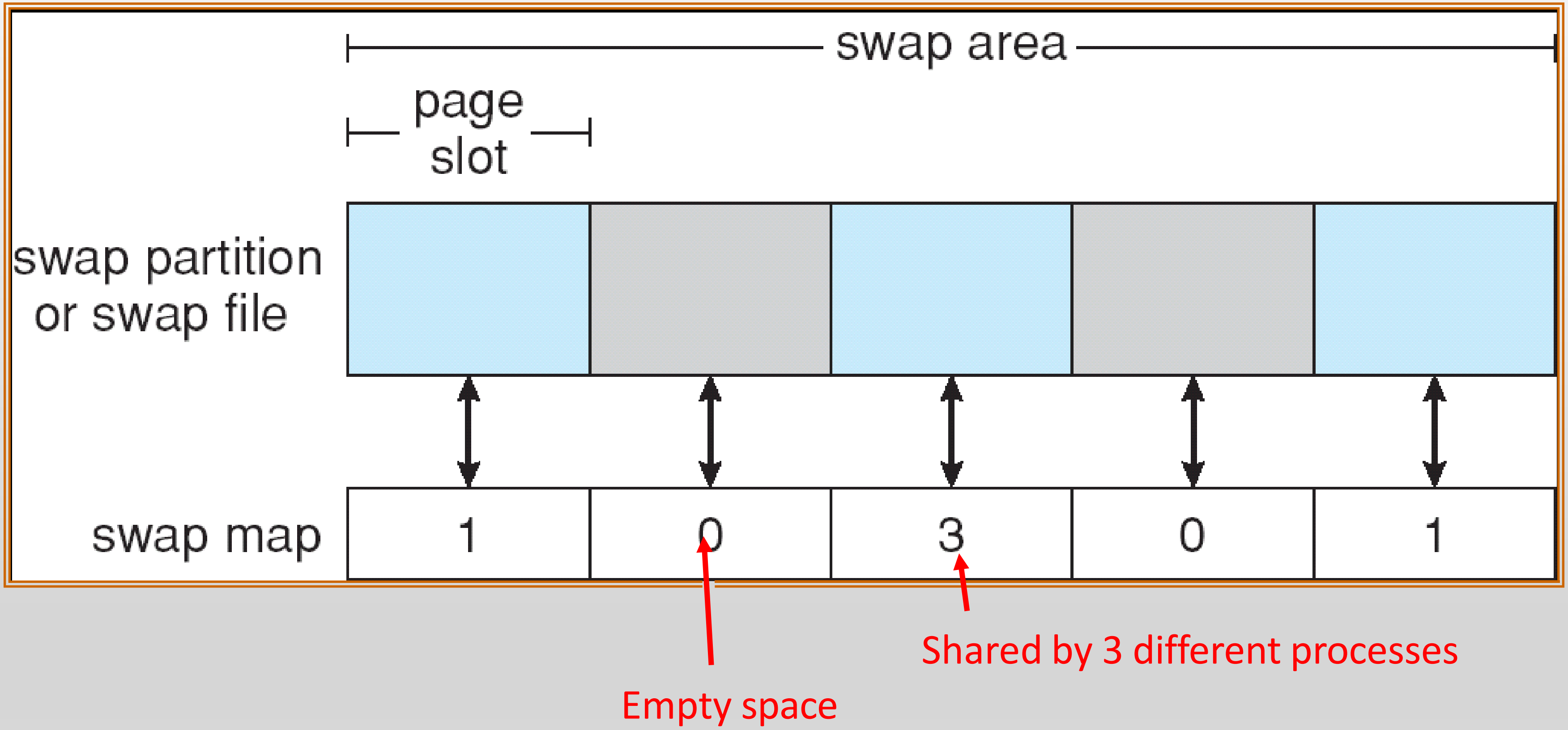
Swap-Space Management

- 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

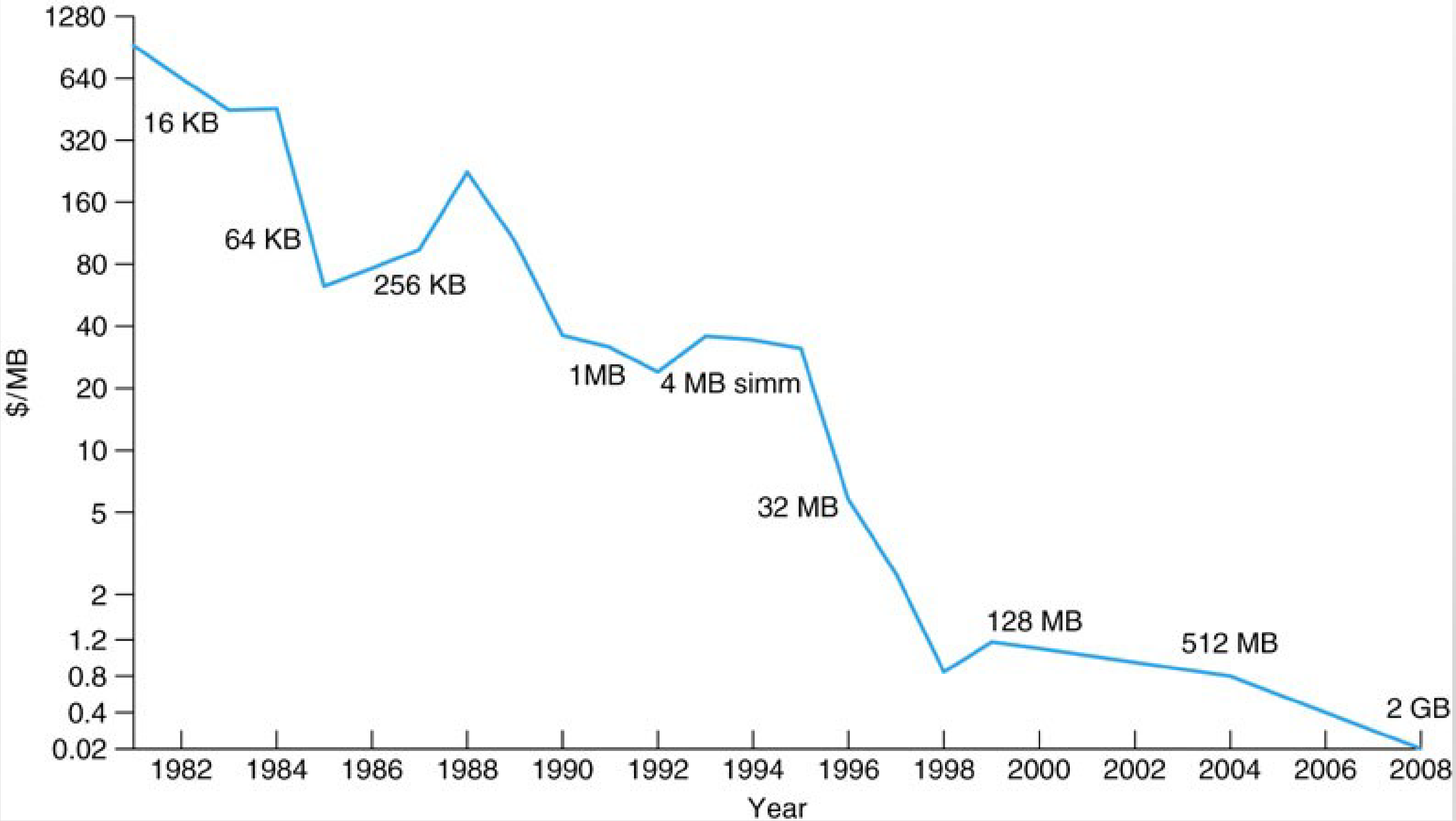
- 1st version: copy entire process between contiguous disk regions and memory
- 2nd version: copy pages to swap space
- Solaris 1:
  - text segments read from file system, thrown away when pageout
  - Only anonymous memory (stack, heap, etc) store in swap space
- Solaris 2:
  - swap-space allocation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

Data Structures for Swapping (Linux)

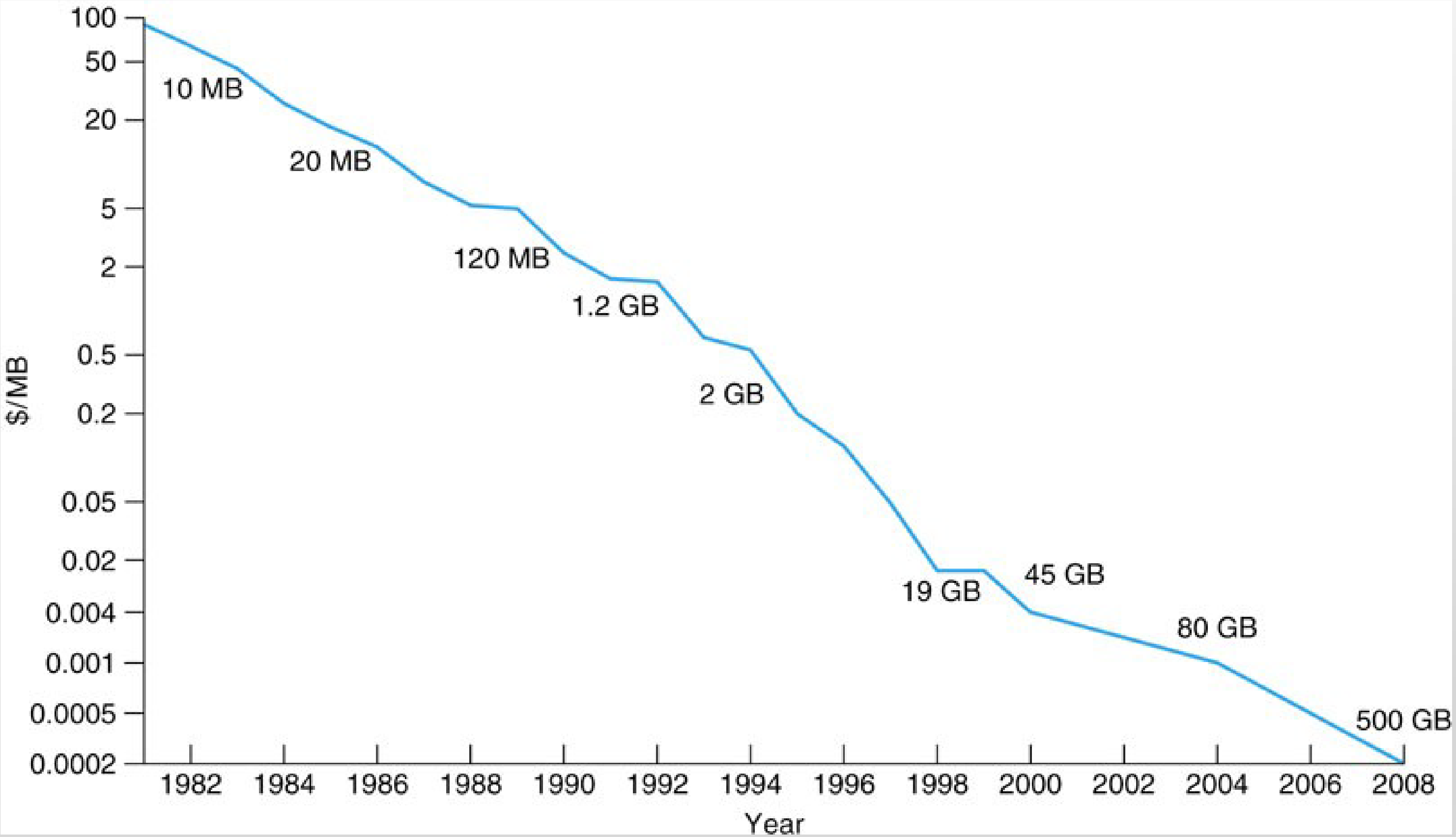


RAID Structure

DRAM Price (1981 - 2008)



Magnetic Hard Disk Price (1981 - 2008)

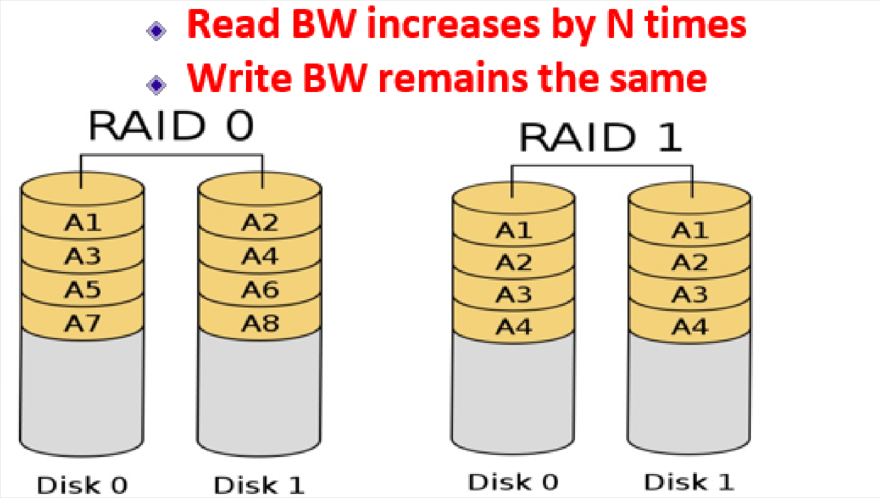


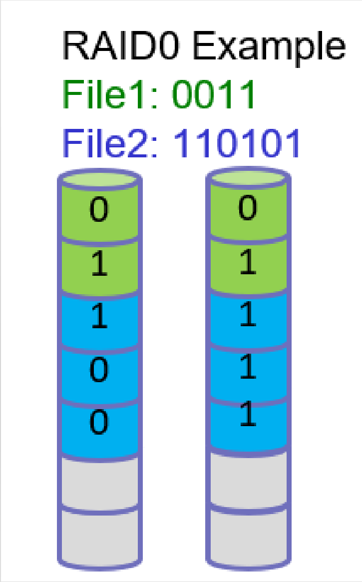
RAID Disks

- RAID = Redundant Arrays of Inexpensive Disks
  - 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

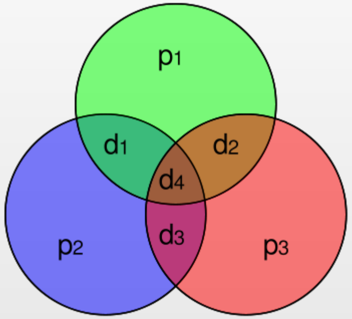
- RAID 0: non-redundant striping
  - Improve performance via parallelism
  - I/O bandwidth is proportional to the striping count
    - Both read and write BW increase by N times (N is the number of disks)
- RAID 1: Mirrored disks
  - Provide reliability via redundancy

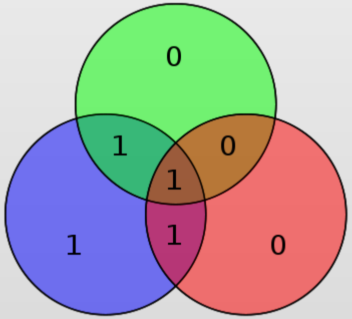


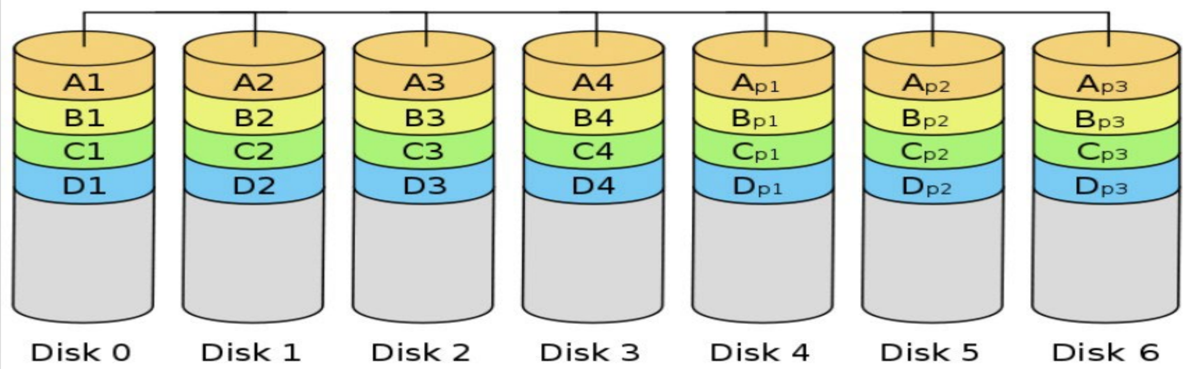


RAID 2: Hamming code

- 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
- Good: Recover from any single disk failure
  - Can detect up to two disks(i.e. bits) error
  - But can only “correct” one bit error
- Good: Better space efficient than RAID1 (75% overhead)

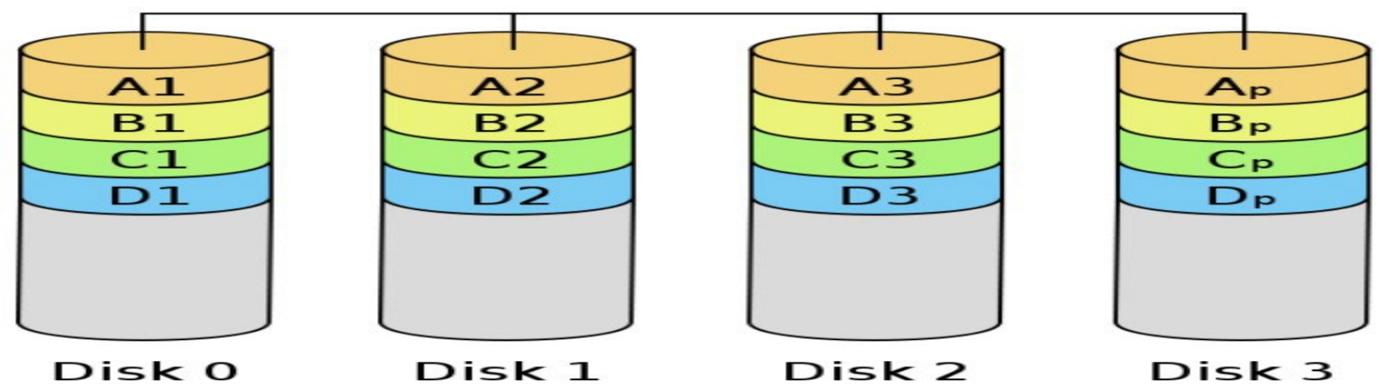






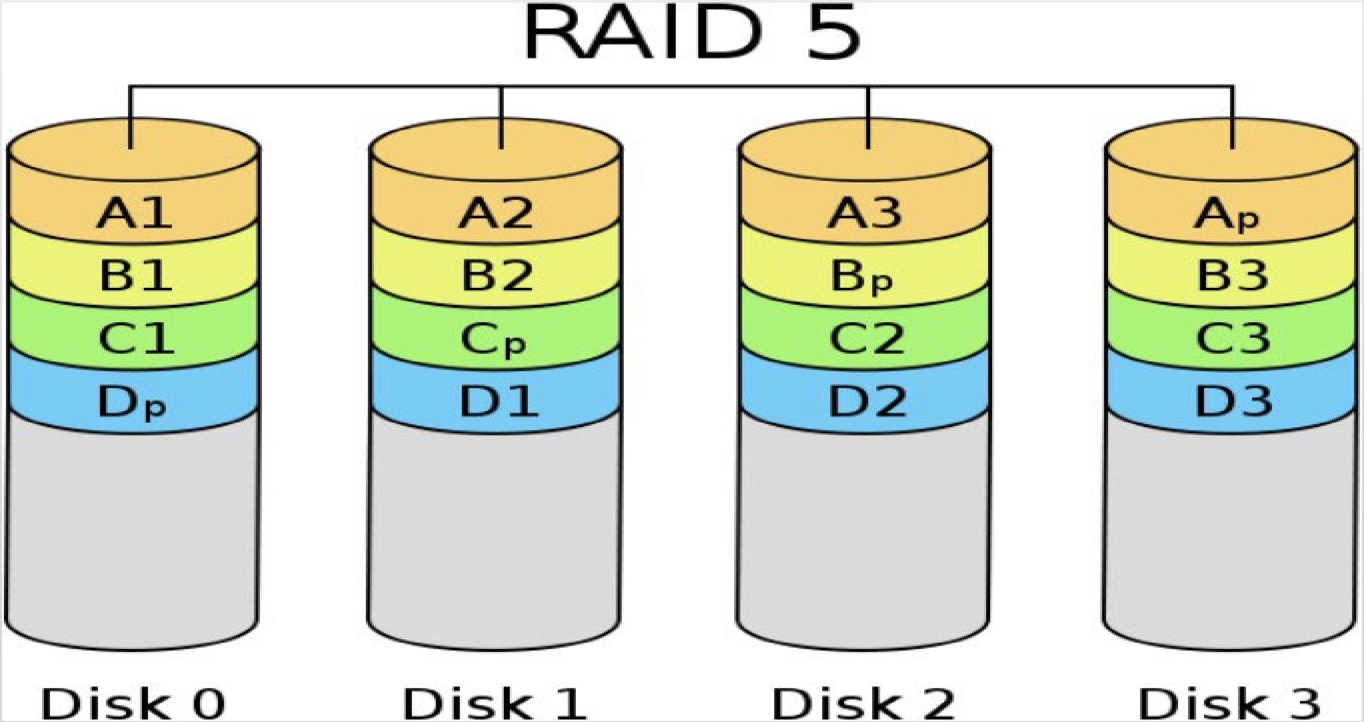
RAID 3 & 4: Parity Bit

- 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
- Good: Even better space efficiency (33% overhead)
- Bad: 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)

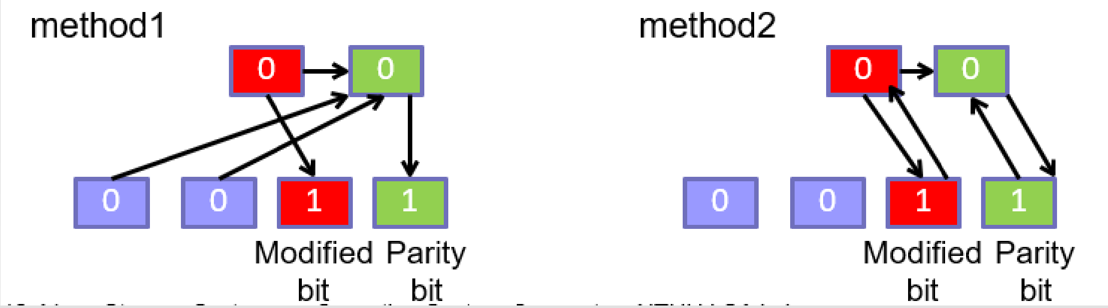


RAID 5: Distributed Parity

- Read BW increases by N times, because all four 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.

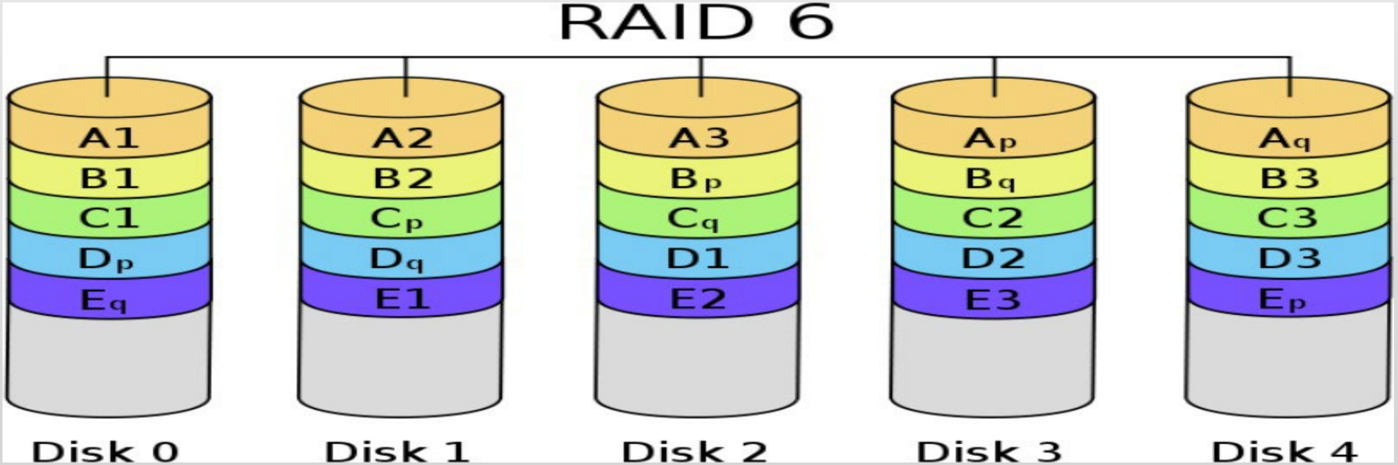
    -> write BW = N / ((N-2)+2) = 1 -> remains the same
  - Method2: (1)only read the parity bit and modified bit. (2) re-compute parity bit by the difference. (3) write both modified bit and parity bit.

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



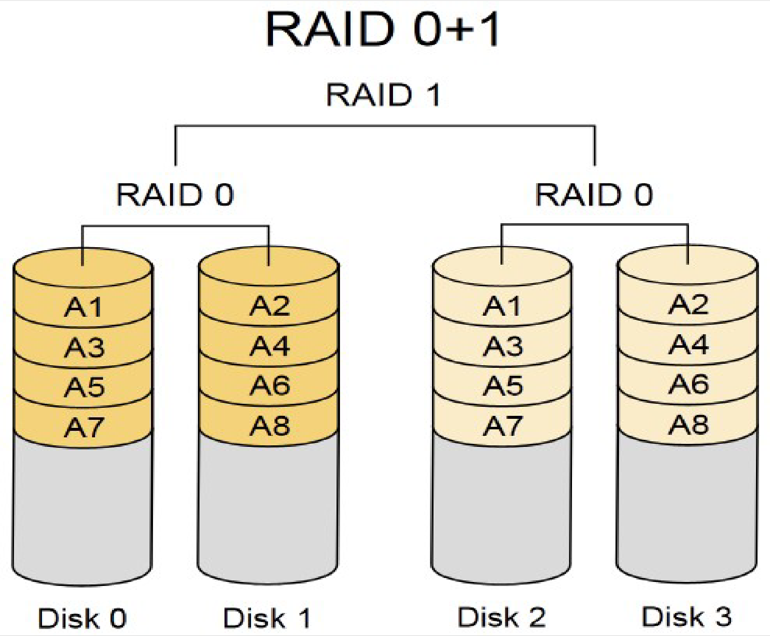
RAID 6: P+Q Dual Parity Redundancy

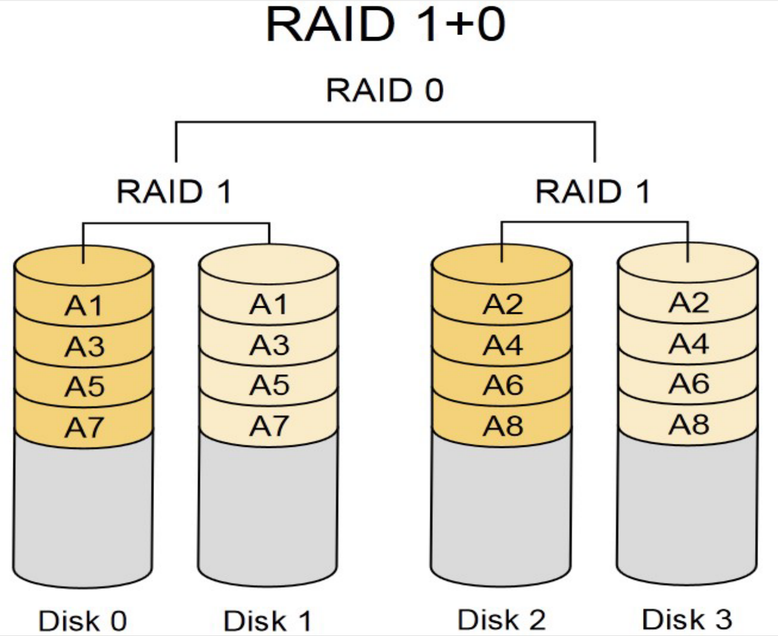
- Like RAID 5, but stores extra redundant information to guard against multiple disk failure
- Use ECE code (i.e. Error Correction Code) instead of single parity bit
- Parity bits are also striped across disks



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

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

Reading Material & HW

- Chapter 12
- Problem Set
  - 12.1
  - 12.3
  - 12.8

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

Q & A

Thank you for your attention