Storage Management

- File System Interface – Chapter 11
- File System Implementation – Chapter 12
- Mass Storage Structure – Chapter 10
- Input-Output Systems – Chapter 13

- NOTE: We are covering this topic in slightly different order than in textbook

Chapter 10: Mass-Storage Structure

- Disk Structure
- Disk Scheduling
- Disk Management
- Swap-Space Management
- Disk Reliability
- Stable-Storage Implementation
- Tertiary Storage Devices
- Operating System Issues
- Performance Issues

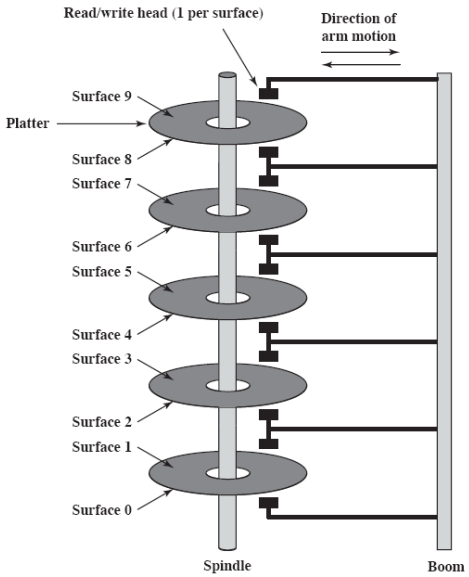
Overview of Mass Storage Structure

- Magnetic disks provide bulk of secondary storage of modern computers
  - Drives rotate at 60 to 200 times per second
    - constant rotational speed (*constant angular velocity*)
  - **Transfer rate** is rate at which data flows between drive and computer
  - **Positioning time** (**random-access time**) is time to move disk arm to desired cylinder (**seek time**) and time for desired sector to rotate under the disk head (**rotational latency**)
  - **Head crash** results from disk head making contact with the disk surface
    - That's really bad
- Disks can be removable
- Drive attached to computer via **I/O bus**
  - Busses vary, including **EIDE, ATA, S-ATA, USB, Fibre Channel, SCSI**
  - **Host controller** in computer uses bus to talk to **disk controller** built into drive or storage array

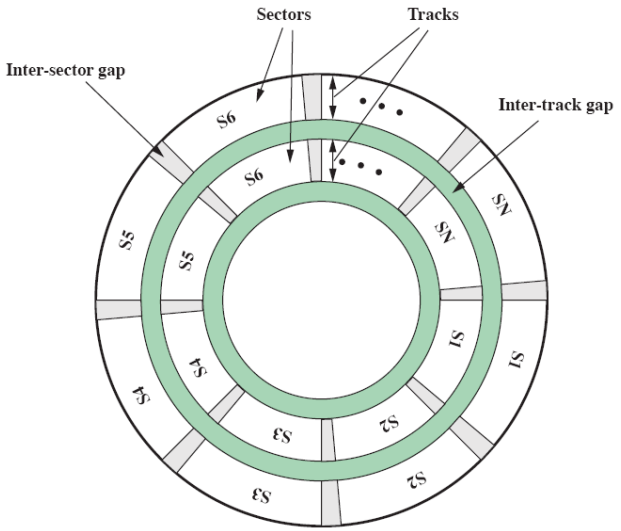
Disk Structure

- Disk drives are addressed as large 1-dimensional arrays of *logical blocks*, where the logical block is the smallest unit of transfer.
- Data blocks on the disk are sectors of storage area laid out according to physical characteristics of the disk device.
- 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.

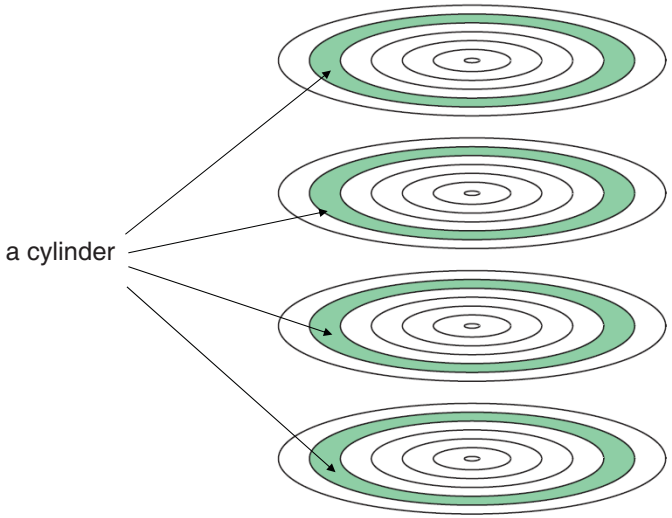
Disk Drive Components



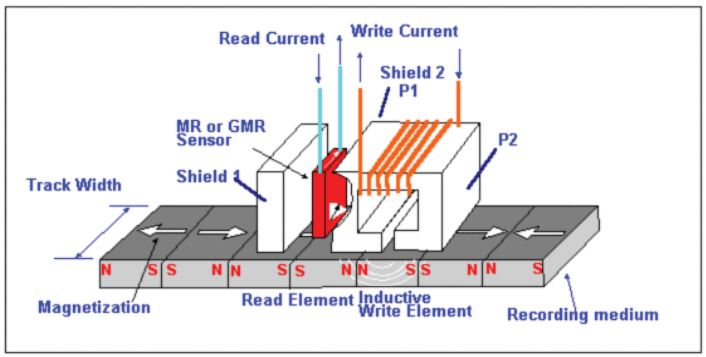
Disk Data Layout



Tracks and Cylinders



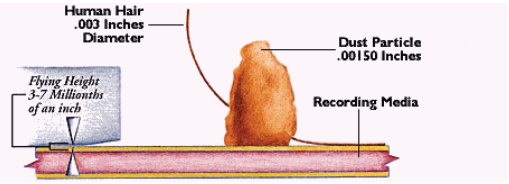
Magnetic Disc R/W Head



Inductive Write - MagnetoResistive Read

Flying Read/Write Head

- Head is aerodynamic and floats on a thin layer of moving air caused by rotation of disk
  - layer of air can be a little as 3 nanometers
- Requires sealed unit to remove particles from air that could cause head to "crash"



Hard Disks

Disk Metrics

- Capacity
  - (platter-surfaces) X (tracks/platter-surfaces) X (sectors/track X byte/sector)
- Seek Time
  - Time to move the arm (heads) to the proper track
- Rotational Latency
  - Time for desired sector to rotate to the head
- Transfer Time
  - Time to read/write a sector of data

Disk Metrics

- 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

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

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

Solid-State Disks

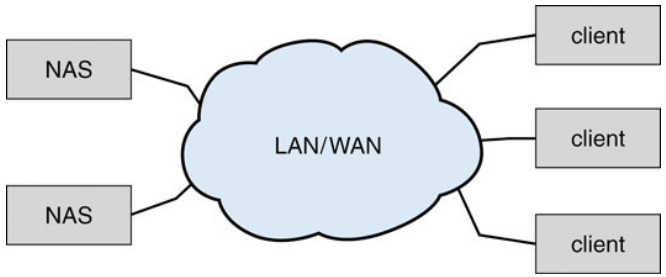
- Nonvolatile memory used like a hard drive
  - Many technology variations
- Can be more reliable than HDDs
- More expensive per MB
- Maybe have shorter life span
- Less capacity
- But much faster
- Busses can be too slow -> connect directly to PCI for example
- No moving parts, so no seek time or rotational latency

Disk Attachment

- Host-attached storage accessed through I/O ports talking to I/O channels (specialized I/O processors) and I/O buses
- Examples of High Performance I/O Buses
  - **Serial ATA (SATA)**
    - ❄ data transmitted as serial packets
  - **Small Computer System Interface (SCSI)** is a I/O bus, up to 16 devices on one cable, SCSI initiator requests operation and SCSI targets perform tasks
    - ❄ Each target can have up to 8 **logical units** (disks attached to device controller
  - **Fibre Channel** is high-speed serial architecture
    - ❄ Can be switched fabric with 24-bit address space – the basis of **storage area networks** (**SAN**s) in which many hosts attach to many storage units
    - ❄ Can accomodate 126 devices

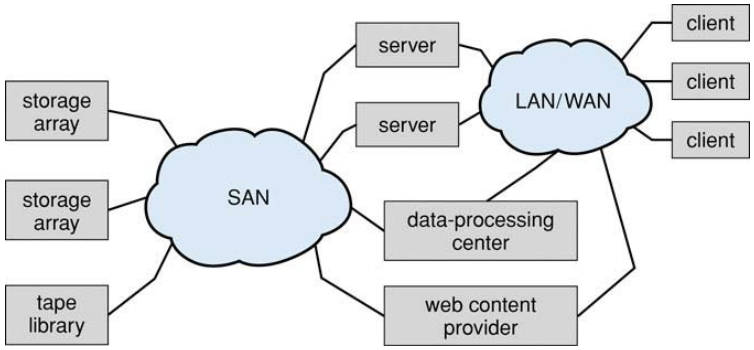
Network-Attached Storage

- Network-attached storage (**NAS**) is storage made available over a network rather than over a local connection (such as a bus)
- NFS and CIFS are common protocols
- Implemented via remote procedure calls (RPCs) between host and storage
- New iSCSI protocol uses IP network to carry the SCSI protocol



Storage Area Network

- Common in large storage environments (and becoming more common)
- Multiple hosts attached to multiple storage arrays - flexible



Disk Scheduling

- The operating system is responsible for using hardware efficiently — for the disk drives, this means considering access time and disk data transfer rate.
- Access time has two major components
  - *Seek time* is the time for the disk to move the heads to the cylinder containing the desired sector.
  - *Rotational latency* is the additional time waiting for the disk to rotate the desired sector to the disk head.
- Want to minimize access time
- Seek time ≈ seek distance
- Overall disk data rate (*bandwidth or throughput*) 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.

Disk Scheduling (Cont.)

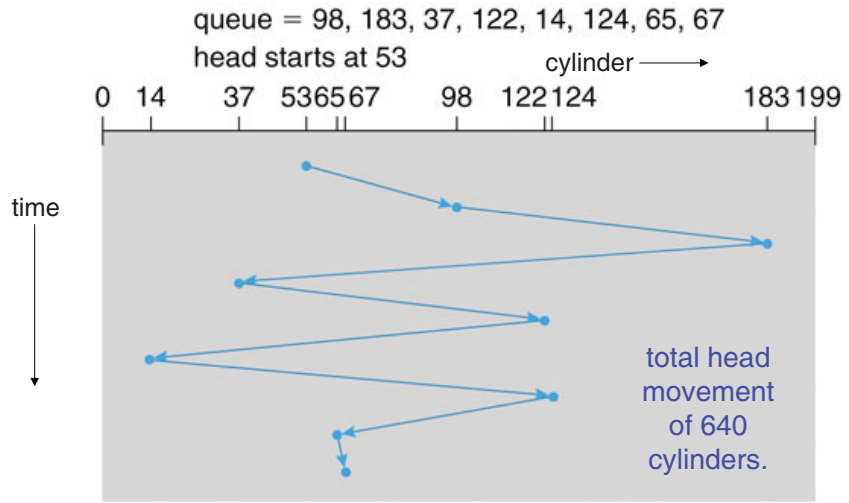
- Several algorithms are commonly used to schedule the servicing of disk I/O requests.
- We illustrate them with a series of access requests to specific cylinders (request queue for cylindrs 0-199).

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

cylinders

Head pointer starts at cylinder 53

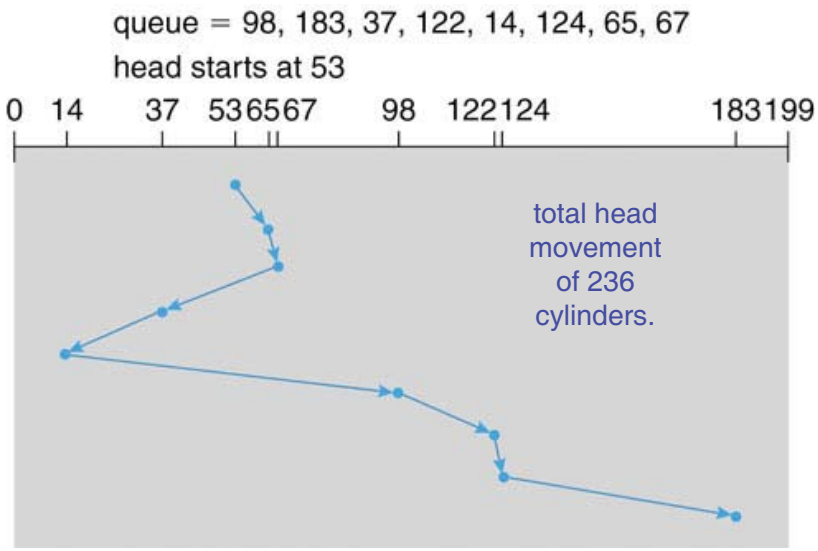
FCFS – First Come First Served



SSTF – Shortest Seek Time First

- Selects the request with the minimum seek time from the current head position.
- SSTF scheduling is a form of the SJF scheduling discussed previously for processes
  - may cause starvation of some requests.

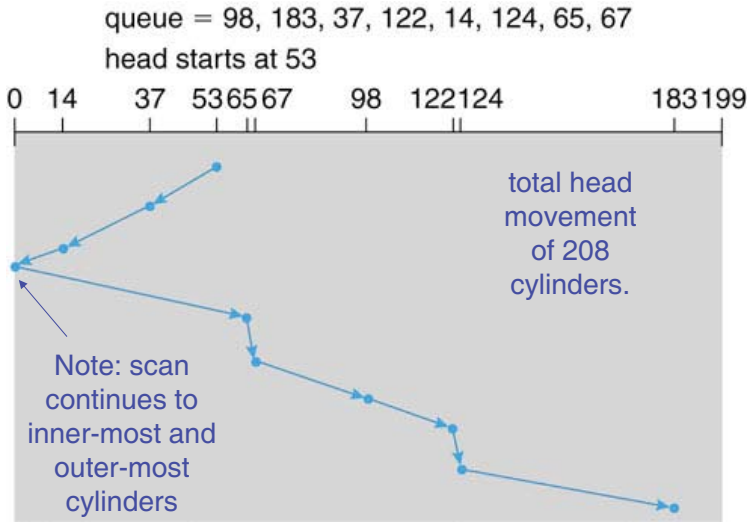
SSTF (Cont.)



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.
- Sometimes called the *elevator algorithm*.
- Illustration shows total head movement of 208 cylinders.

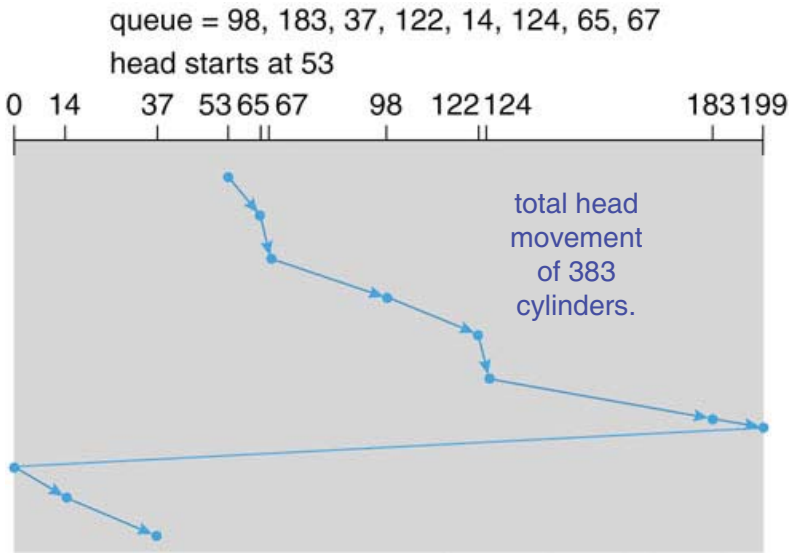
SCAN (Cont.)



C-SCAN – Circular 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.

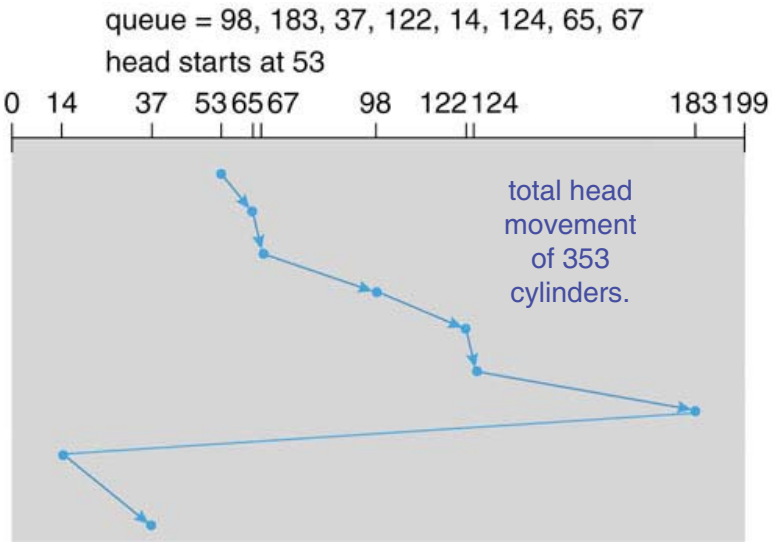
C-SCAN (Cont.)



C-LOOK (look ahead)

- Version of C-SCAN
- Arm only goes as far as the last request in each direction, then reverses direction immediately, without first going all the way to the end of the disk.
- Same variation applied to SCAN algorithm called LOOK

C-LOOK (Cont.)

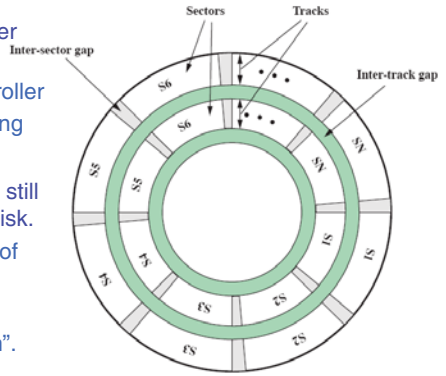


Selecting a Disk-Scheduling Algorithm

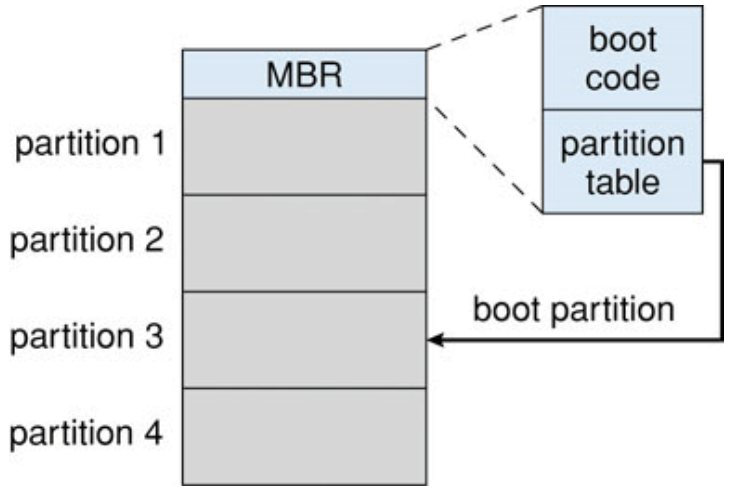
- SSTF is common and has a natural appeal
- SCAN and C-SCAN perform better for systems that place a heavy load on the disk.
  - probably use LOOK and C-LOOK in this situation
- Performance depends on the number and types of requests.
- Requests for disk service can be influenced by the file-allocation method.
- The disk-scheduling algorithm should be written as a separate module of the operating system, allowing it to be replaced with a different algorithm if necessary.
- Either SSTF or LOOK is a reasonable choice for the default general-purpose algorithm.

Disk Management

- *Low-level formatting*, or *physical formatting* — Dividing a disk into sectors that the disk controller can read and write.
  - Sector Formatting used by hardware controller
  - header, data, trailer (used for error detecting and correcting)
- 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.
  - initialize directory
  - *Logical formatting* or "making a file system".
- A boot block can be stored on disk to be used in initializing the operating system.
  - The *bootstrap loader* program (stored in ROM firmware) loads the boot block.
- Methods such as *sector sparing* can be used to handle bad blocks.



Booting from a Disk in Windows 2000

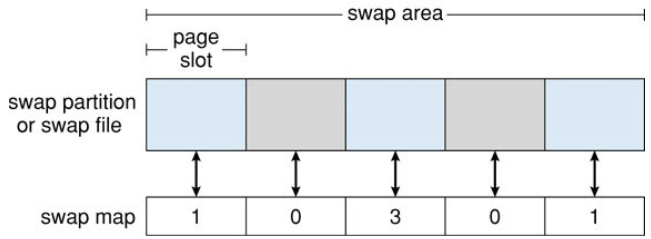


Swap-Space Management

- Swap-space — Virtual memory uses disk space as an extension of main memory.
- Swap-space can be carved out of the normal file system or, more commonly, it can be in a separate disk partition.
- Swap-space management
  - older 4.3BSD Unix allocates swap space for the entire process image when process starts.
    - ❄ Kernel uses *swap maps* to track swap-space use.
  - newer Solaris 2 allocates swap space only when a page is forced out of physical memory, not when the virtual memory page is first created.
    - ❄ read-only pages are never swapped out – just read again from file system when needed.
      - swap space used only for *anonymous* memory (stack, heap, etc.)

Data Structures for Swapping on Linux Systems

- similar to Solaris 2
  - swap space used only for anonymous memory (stack, heap, etc) or shared memory
  - normal file system can be used for swap area or a swap partition can be used
  - swap area divided in *slots* of page size
  - swap map tracks availability of slots



Swap Map Values: 0 – available, otherwise how many processes map to that page

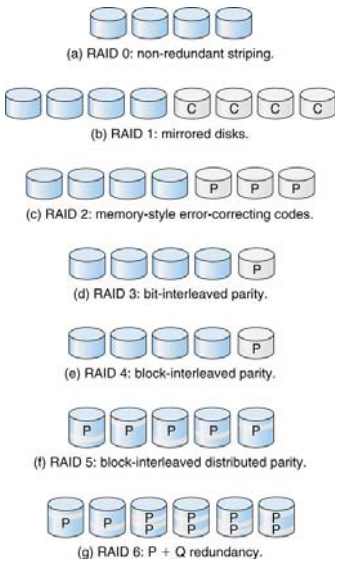
Disk Reliability & Performance

- Several improvements in disk-use techniques involve the use of multiple disks working cooperatively.
- Disk striping uses a group of disks as one storage unit.
  - Each data block broken into multiple strips
  - One strip per disk (a *stripe* is a set of strips that define block)
  - Performance enhanced – no reliability improvement
- RAID schemes improve performance and improve the reliability of the storage system by storing *redundant* data.
  - *Mirroring* or *shadowing* keeps duplicate of each disk.
  - *Block interleaved parity* uses much less redundancy
    - Example – 9 disks (8 data blocks and 1 parity block)
      - bit level strips

RAID

- Redundant Array of Independent Disks
  - Redundant Array of Inexpensive Disks - ??
- 7 different configurations defined
- Not a hierarchy
  - different configurations for different applications
- Set of physical disks viewed as single logical drive by O/S
  - hardware provides a RAID controller
- Data distributed across physical drives
- Can use redundant capacity to store parity information
  - allows error detection and correction

RAID Levels



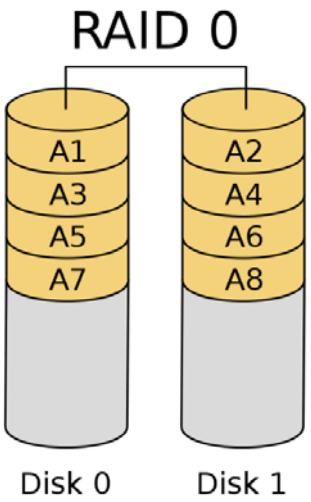
Data Striping

- Data striping (splitting) is the way that logically sequential data is mapped to multiple storage units
- Can be useful for several reasons:
  - performance increase
    - read
    - write
    - both read and write
  - reliability
    - need to store redundant information
    - data recovery in case of hardware failure or read/write errors
- A particular configuration may increase performance or increase reliability or both

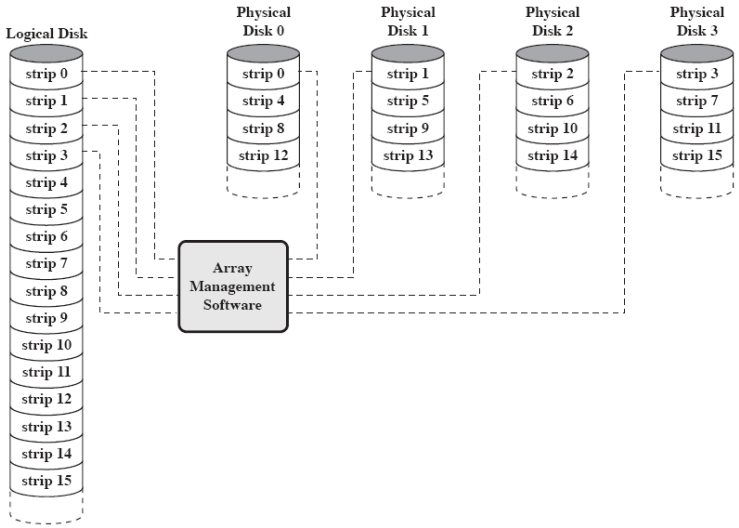
RAID 0

- No redundancy
- Data striped across all disks
- Round Robin striping
- Used to increase access speed
  - Multiple data requests probably not on same disk
  - Disks seek in parallel
  - A set of data is likely to be striped across multiple disks
- One disk failure will cause loss of data

RAID 0



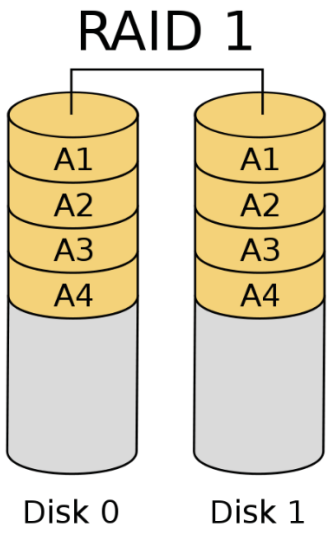
Data Mapping For RAID 0



RAID 1

- Mirrored Disks
- Data is striped across disks
- 2 copies of each stripe on separate disks
- Speed increase for read operations
  - Read from either disk holding same stripe
- Speed of write same as one disk
  - Write to both disks holding same stripe
- Recovery is simple
  - Swap faulty disk & re-mirror
  - No down time
- Expensive
  - requires a lot of disks (twice the desired capacity)

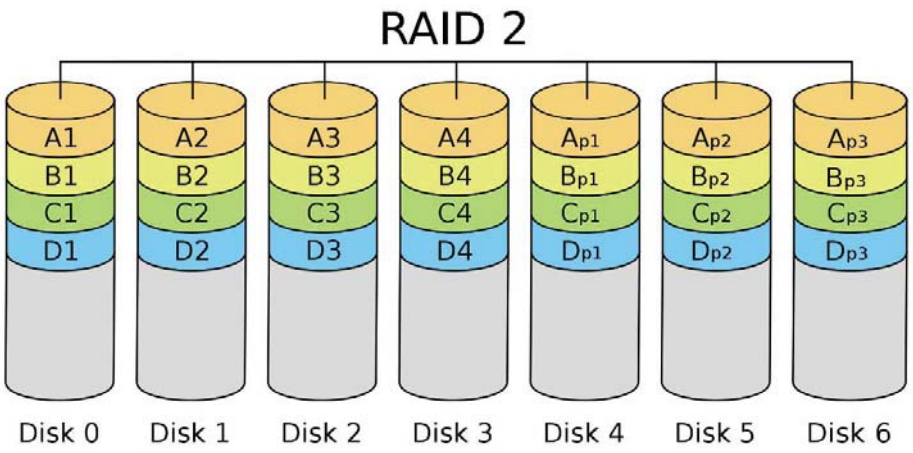
RAID 1



RAID 2

- Very small stripes
  - each stripe is one bit
- Disks are physically synchronized in rotation
  - requires parallel access to all disks for either read or write
- Error correction calculated across corresponding bits on disks
  - Multiple parity disks store Hamming code error correction information
  - possibility of detecting and correcting multiple-bit errors
- Lots of redundancy
  - Expensive
  - Useful if using disks with high error rates
    - Given modern reliable disks - not used
- Very high transfer rates for read and write

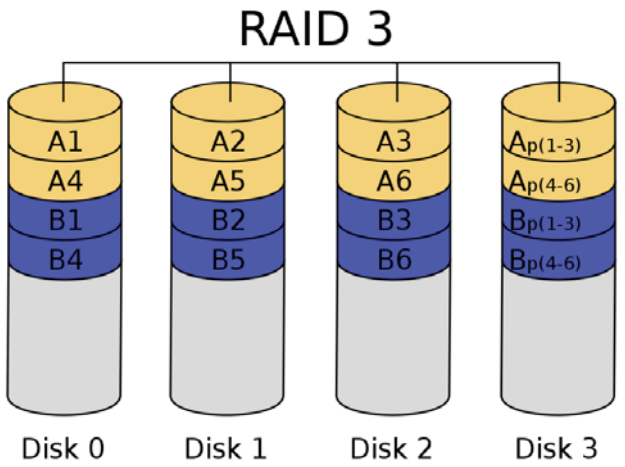
RAID 2



RAID 3

- Similar to RAID 2
- Uses byte-level stripes
- Only one redundant disk, no matter how large the array
- Simple parity strip for each set of corresponding data stripes
- Can detect odd-numbers of byte errors
- Can correct single-byte errors
- Data on one failed drive can be reconstructed from surviving data and parity information
- Very high transfer rates for read and write
- Also requires spin synchronization
- Not commonly used

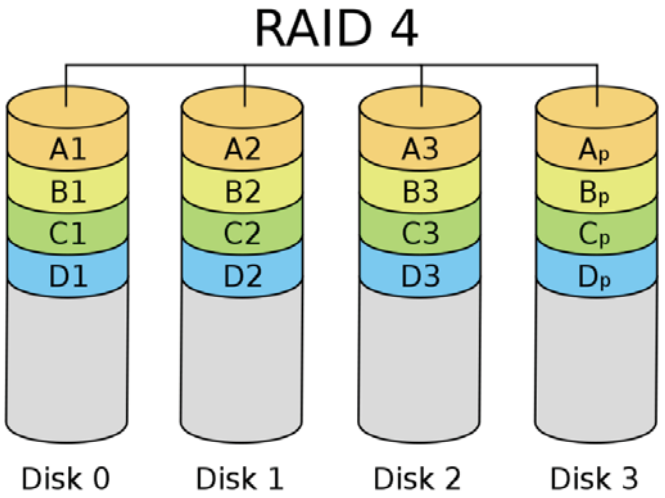
RAID 3



RAID 4

- Each disk operates independently
- Large stripes
  - many bytes in a block
  - one block per strip
- Good for high I/O request rate
  - can perform parallel reads and writes on sequential blocks
- Bit by bit parity calculated across stripes on each disk
- Parity stored on one parity disk
  - each access to a block also requires access to parity disk
- Can survive one disk failure
- Also not commonly used

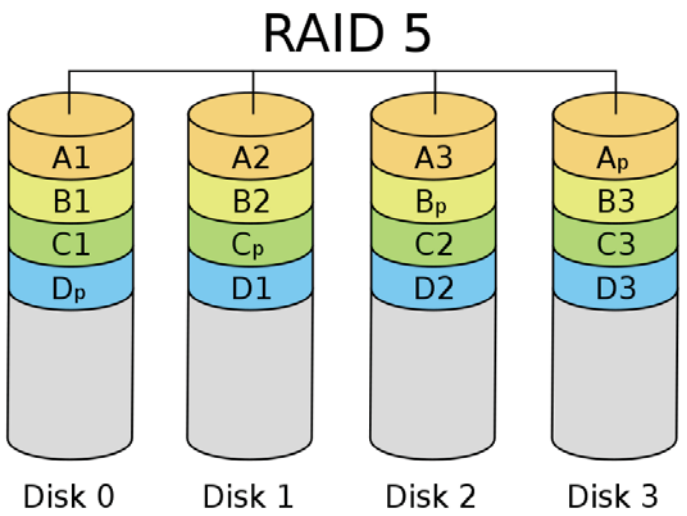
RAID 4



RAID 5

- Similar to RAID 4
- No one dedicated parity disk
  - Parity stripped across all disks
- Round robin allocation for parity strip
- Avoids RAID 4 bottleneck at parity disk
- Commonly used in network servers

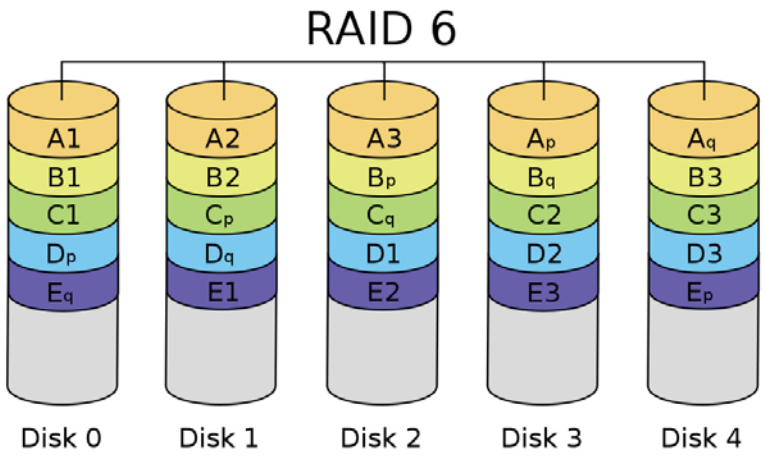
RAID 5



RAID 6

- Two parity calculations used
  - Stored in separate blocks on different disks
- User requirement of N disks will require N+2 disks to implement
- More complex to implement
- High data availability
  - can operate with two disk failures
  - Three disks need to fail for data loss
    - third disk would have to fail while rebuilding in process to correct two previous disk failures
  - Significant write penalty
    - have to write two parity blocks

RAID 6



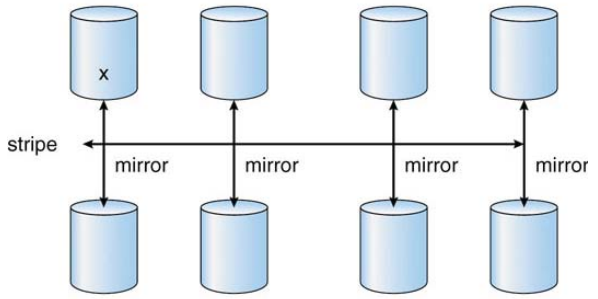
RAID Summary

Category	Level	Description	Disks Required	Data Availability	Large I/O Data Transfer Capacity	Small I/O Request Rate
Striping	0	Nonredundant ★	N	Lower than single disk	Very high	Very high for both read and write
Mirroring	1	Mirrored ★	$2N$	Higher than RAID 2, 3, 4, or 5; lower than RAID 6	Higher than single disk for read; similar to single disk for write	Up to twice that of a single disk for read; similar to single disk for write
Parallel access	2	Redundant via Hamming code	$N + m$	Much higher than single disk; comparable to RAID 3, 4, or 5	Highest of all listed alternatives	Approximately twice that of a single disk
	3	Bit-interleaved parity	$N + 1$	Much higher than single disk; comparable to RAID 2, 4, or 5	Highest of all listed alternatives	Approximately twice that of a single disk
Independent access	4	Block-interleaved parity	$N + 1$	Much higher than single disk; comparable to RAID 2, 3, or 5	Similar to RAID 0 for read; significantly lower than single disk for write	Similar to RAID 0 for read; significantly lower than single disk for write
	5	Block-interleaved distributed parity ★	$N + 1$	Much higher than single disk; comparable to RAID 2, 3, or 4	Similar to RAID 0 for read; lower than single disk for write	Similar to RAID 0 for read; generally lower than single disk for write
	6	Block-interleaved dual distributed parity	$N + 2$	Highest of all listed alternatives	Similar to RAID 0 for read; lower than RAID 5 for write	Similar to RAID 0 for read; significantly lower than RAID 5 for write

N = number of data disks; m proportional to log N

★ - commonly used

RAID (1 + 0), aka RAID 10



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

Stable-Storage Implementation

- Write-ahead log (transaction processing) scheme requires *stable storage* for the log.
  - Stable storage must never be lost
- To implement stable storage:
  1. Replicate information on more than one nonvolatile storage media with independent failure modes.
  2. Update information in a controlled manner to ensure that we can recover the stable data after any failure during data transfer or recovery.

Tertiary Storage Devices

- Low cost is the defining characteristic of tertiary storage.
- Generally, tertiary storage is built using *removable media*
- Common examples of removable media are magnetic tape, floppy disks, CD-ROMs and data DVDs
  - a variety of other types are available.