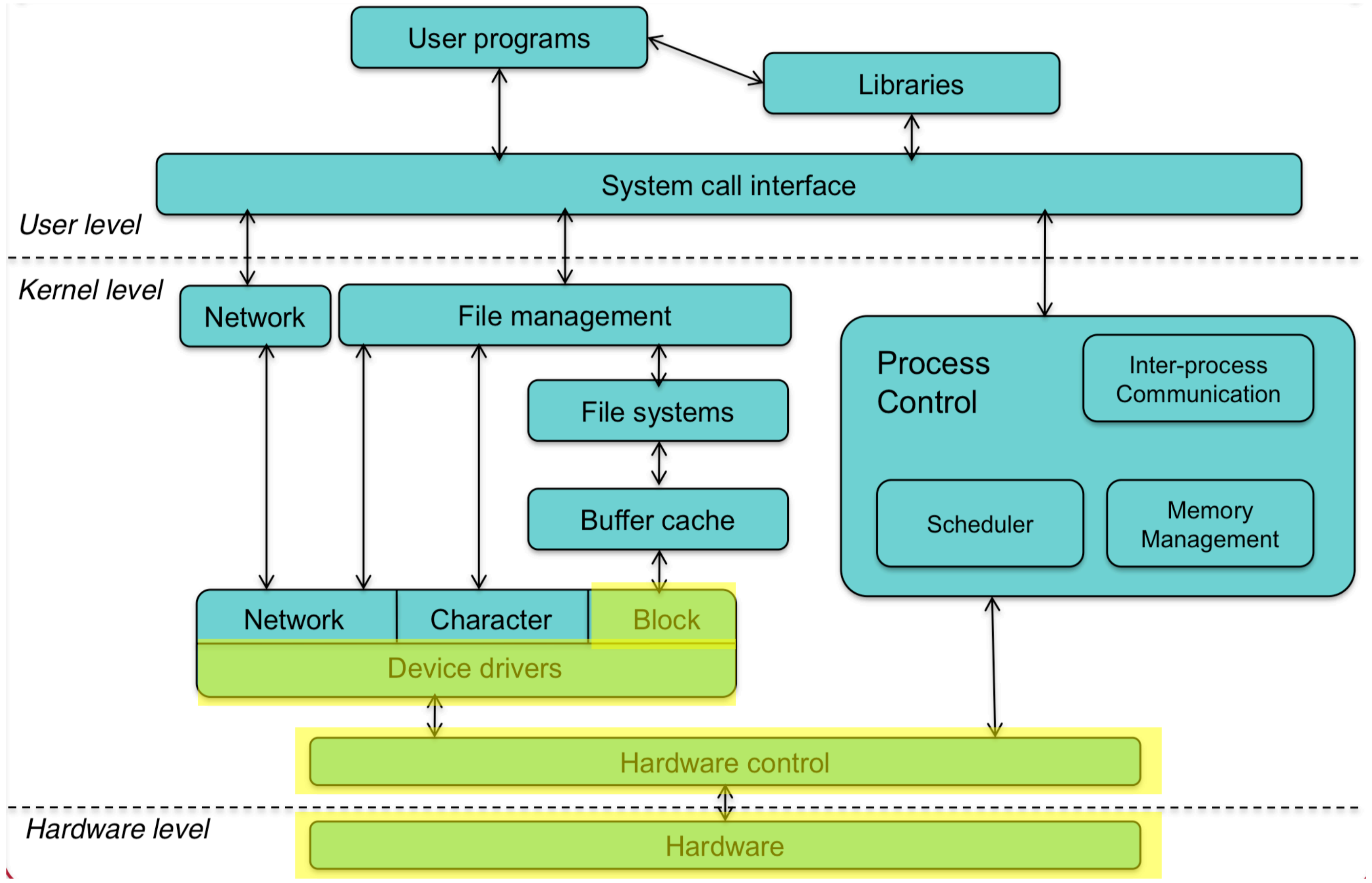
Computers & Operating Systems

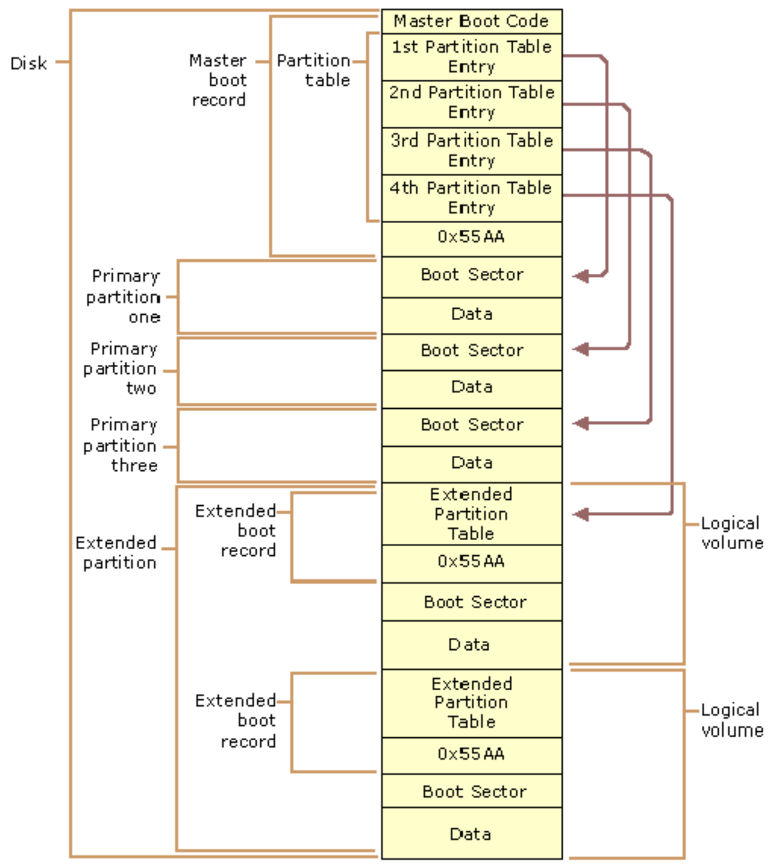
LS-03. OS Booting. Mass-Storage Systems.



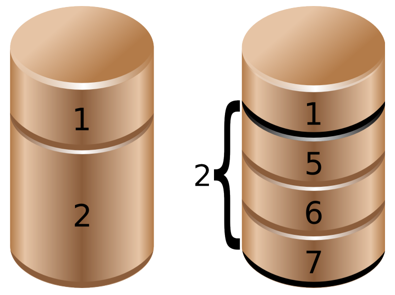
Agenda

- 1. OS Booting
  - Disk Partition MBR
  - Disk Partition GPT
  - OS Boot Process: Linux
  - UNIX/Linux Run Levels
- 2. Mass-Storage Systems
  - Overview of Mass Storage Structure
  - Disk Structure
  - Disk Attachment
  - Disk Management
  - Disk Scheduling
  - Swap-Space Management
  - RAID Structure
  - Stable-Storage Implementation

1. OS Booting

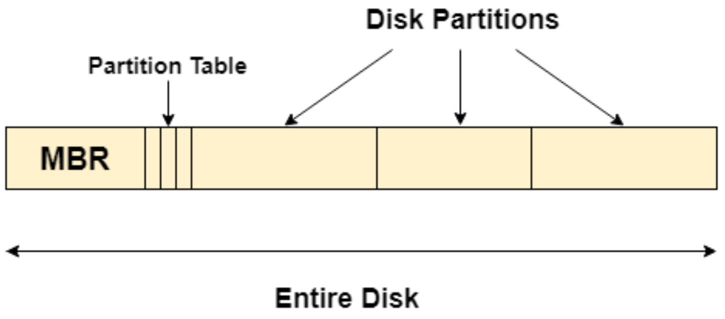


- Disk Partition MBR
- Disk Partition GPT
- OS Boot Process: Linux
- UNIX/Linux Run Levels



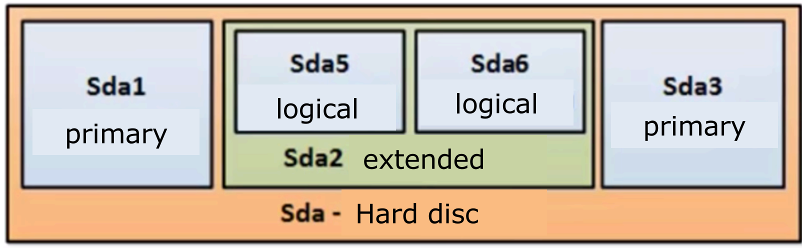
Disc partition: Old – MBR (since 1983)

- Partition the disc with MBR structure (max 4 Primary partitions)

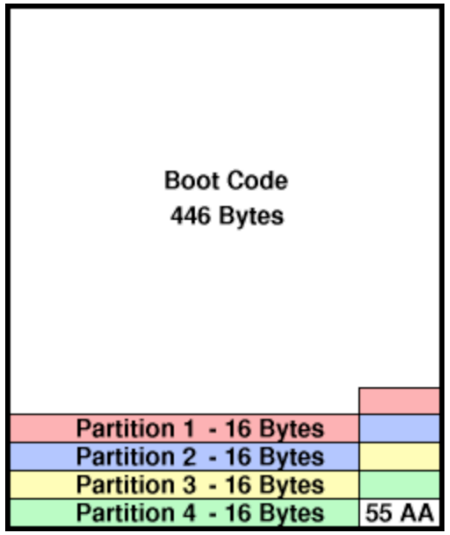


MBR - Master Boot Record

- Numbering disk partitions in Linux (ATA disc – hd; SATA disc – sd)



- MBR structure
  - Mini loader 446 Byte
  - 4 partition 4x16 Byte
  - Magic Nr 2 Byte (55AA)



- MBR Disadvantages.
  - MBR have only 4 boot (primary) partitions
  - MBR have the limited size of 32 bits for block addresses and related information.

For hard disks with 512-byte sectors, the MBR partition table entries allow a maximum size of ($2^{32} \times 512$ bytes) = $2^{32} \times 2^{9} = 2^{41} = = 2 \times 2^{40} = 2$ TiB

Today sale 16 TiB (300 $)

- Commands for disk partition:
  - Linux: fdisk, partition, GParted
  - Mac OS: Disk Utility>Volume>Partition
  - Windows: Administrative Tools >Computer Management >DiskManagement

- OS Loaders
  - NTLDR (WinNT)
  - LILO (Linux,BSD)
  - GRUB (GNU)
  - BootX (MacOS)
  - Chameleon (CrosOS)

Disc partition: New – GPT (since 2005)

- Partition the disc with GUID-UEFI structure
  - GPT - GUID Partition Table
  - GUID - globally unique identifier
  - UEFI -Unified Extensible Firmware Interface
  - Protective MBR - protection against old partitioning tools (compatibility)
- Advantages:
  - BackUp copy of GPT for auto repair
  - Up to 128 partitions (and more).
  - Max Partition Size = 8 ZiB.
  - UEFI created for 32- and 64-bit architectures.

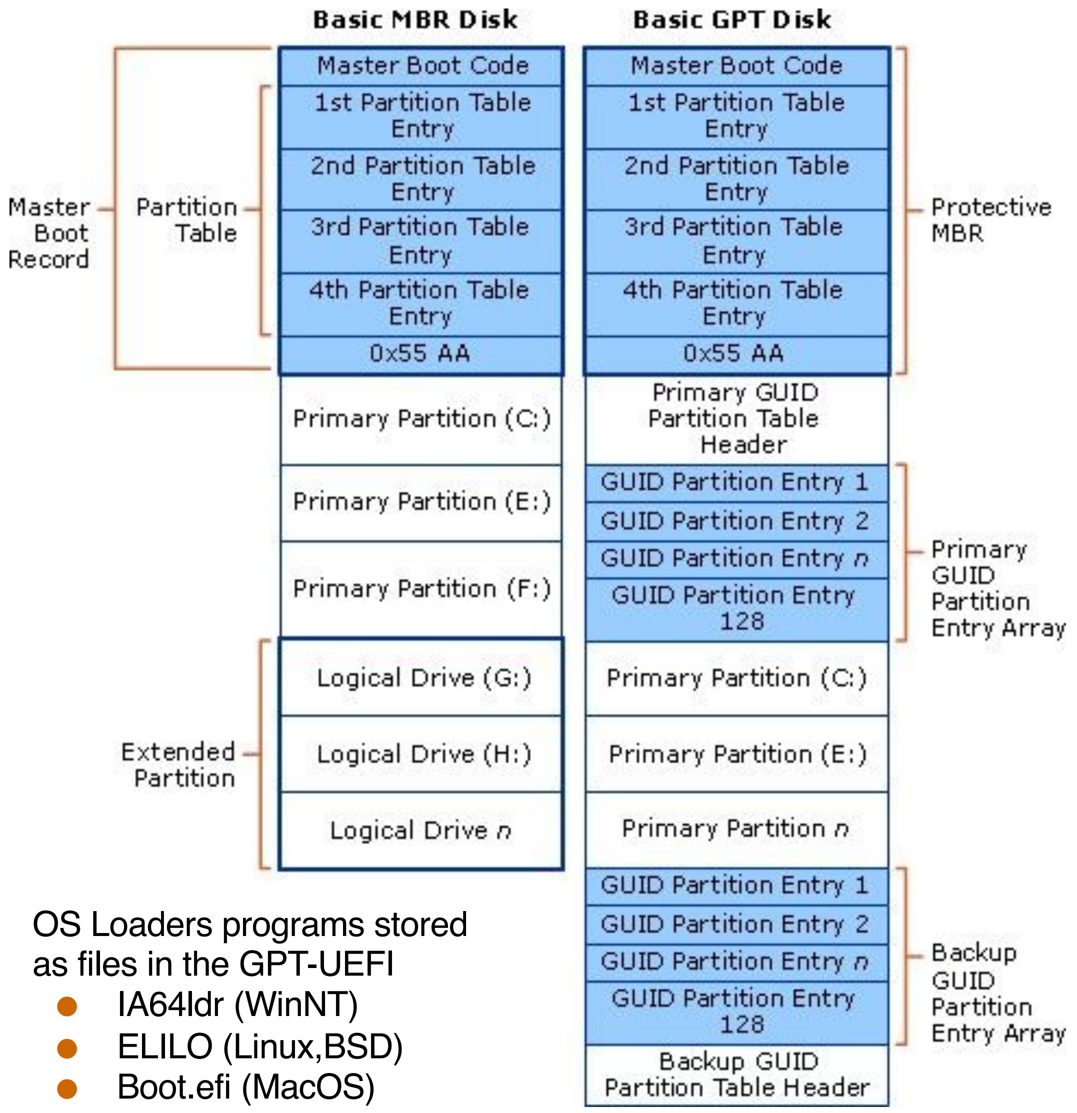
Note:

- GUID size = 128 bit. Example:

{6F9619FF-8B86-D011-B42D-00CF4FC964FF}

- The uniqueness of generation each individual GUID is not guaranteed:

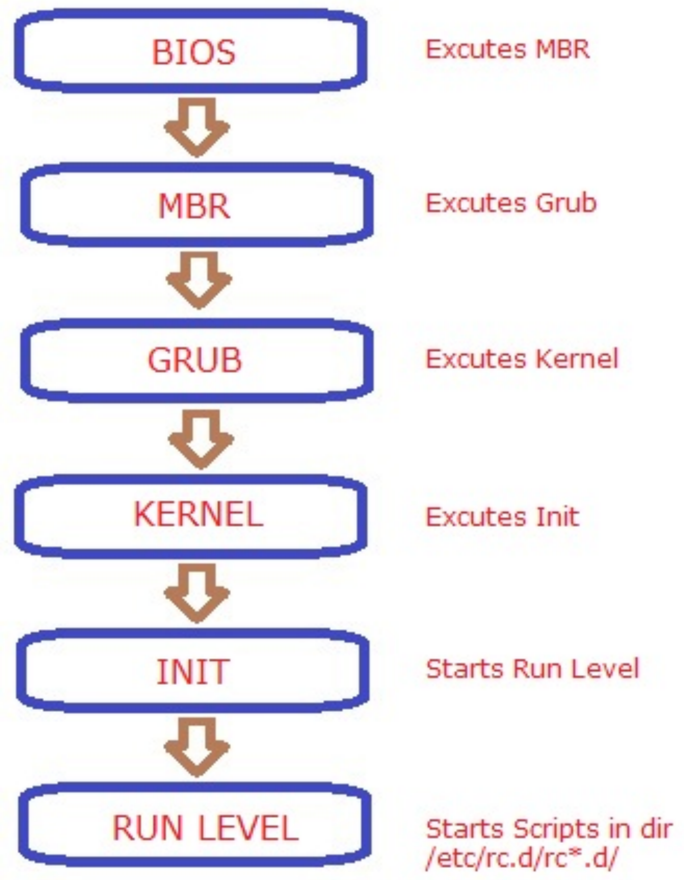
Probability of equality Peq=$(1/2^{128})$*Tsec

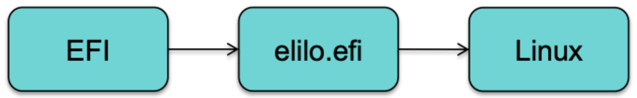


- OS Loaders programs stored as files in the GPT-UEFI
  - IA64ldr (WinNT)
  - ELILO (Linux,BSD)
  - Boot.efi (MacOS)

OS Boot Process: Linux

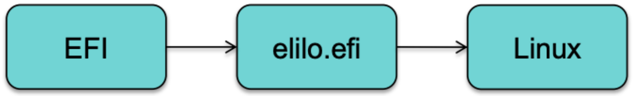
- Linux booting process (or "**boot sequence**") can be divided to 6 levels
- **1a. BIOS** (Base I/O System) or
- **1b. UEFI** for new computer.
  - POST (Power on Self Test) checks the integrity of the following hardware components: Timer, CPU, DMA controllers, Video ROM, Motherboard, Keybrd, HDD, etc.
  - CMOS is a small memory RAM motherboard chip, not clear its memory when a computer is turned off. It remembers all the configuration with the help of a battery called CMOS battery.
  - After select booting device, BIOS load MBR (GPT).
- **2a. MBR** (Master Boot Record).
  - It is located in the 0st sector of the bootable disk. Typically /dev/hda, or /dev/sda.
  - MBR is less than 512 bytes in size. This has 3 components:
    - 1) primary boot loader info in 1st 446 bytes
    - 2) partition table info in next 64 bytes (16x4)
    - 3) MBR validation check in last 2 bytes.
  - It contains information about GRUB (or LILO).
- **2b. GPT**
  - UEFI located 1-33st sectors (use 0 sector for compatibility)
  - UEFI load file in the EFI boot partition (elilo.efi, boot.efi)



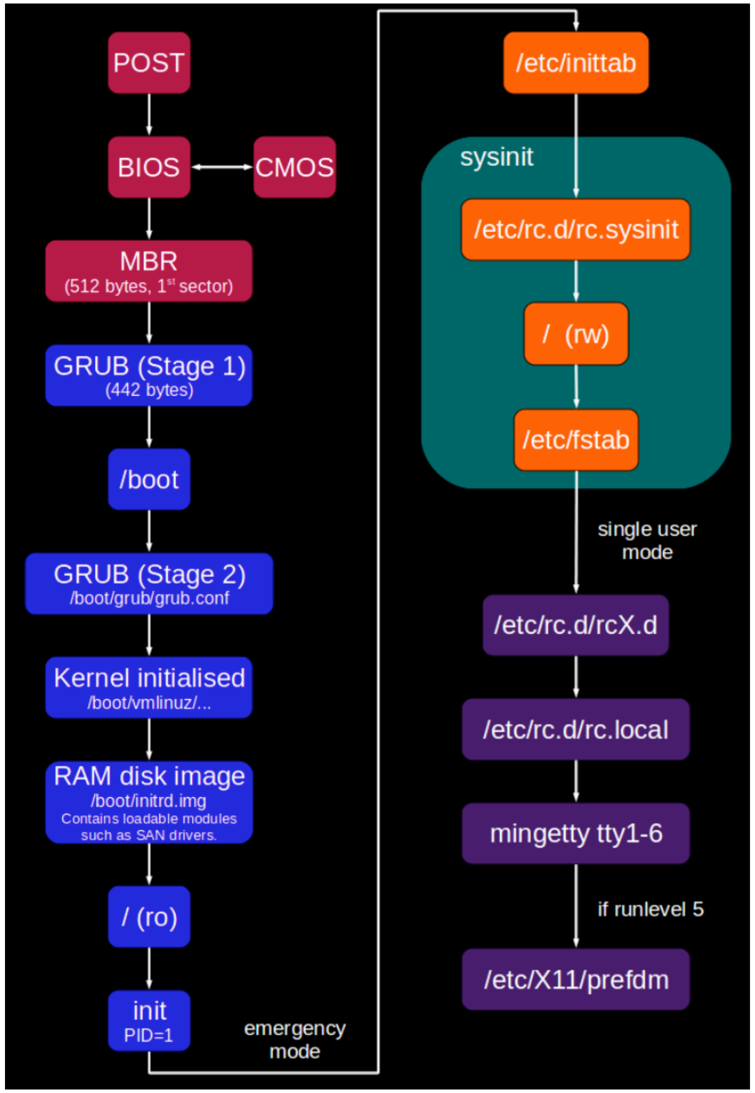


OS Boot Process: Linux (Cont.1)

- **3a. GRUB** (Grand Unified Bootloader).
  - Grub stage 1 will load grub stage 1.5 to the RAM, and will pass the control to it.
  - Grub Stage 1.5 located in the MBR GAP (sector 1 to 63 before the beginning of the first partition) basically contains the drivers for reading file systems.
  - Grub Stage 2 read /boot/grub/grub.conf file which contains other details about kernel path and initrd path etc.
- **3b. ELILO for UEFI**



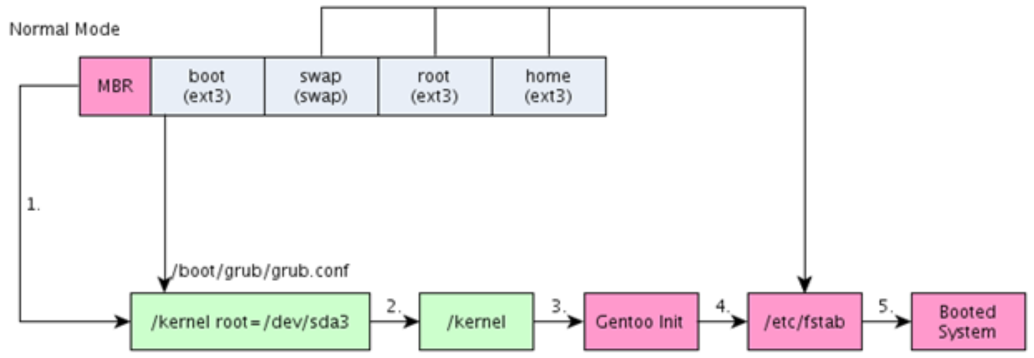
- **4. Kernel**
  - Kernel is a compressed executable bzImage file.
  - It mounts the root file system as specified in the “root=” in grub.conf.
  - Kernel executes the /sbin/init program
  - Since init was the 1st program to be executed by Linux Kernel, it has the process id (PID) of 1.
    Do a ‘ps -ef l grep init’ and check the pid.

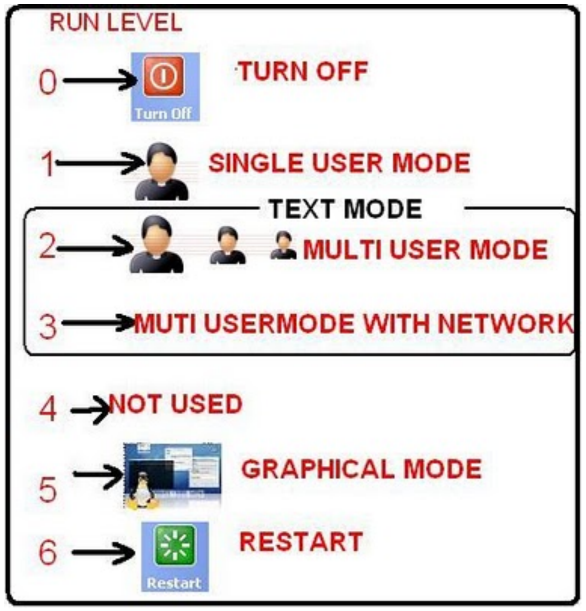


OS Boot Process: Linux (Cont.2)

- **5. Init.**
  - Have 3 styles for init-process: SysV (old), systemd (parallel), upstart (by event). Ubuntu use SysV+upstart.
  - Process /sbin/init reads /etc/fstab file to connect the Linux file systems.
  - Process /sbin/init reads /etc/inittab file to decide the Linux default run level.
  - Set default runlevel (**telinit** command allows admin to change current runlevel).
  - Calls /etc/rc.d/rc.sysinit and /etc/rc.d/rcN/ (where N is a runlevel), example rc5.
- **6. Runlevel.**
  - When the Linux system is booting up, you might see various service getting started. For example, it might say "starting sendmail OK.
  - In /etc/rc.d/rc5.d/ directory starting scripts: with name Knndeamon –> **Kill** sequentially nn, with name SnnDeamon –> **Startup** sequent. nn.
  - Start up the tty processes and xdm (gui-manager)
  - Starts User's login screen
- 7. Modern system boot. Read

https://www.ibm.com/developerworks/linux/library/l-customize-monitor-linux/index.html?S_TACT=105AGX99&S_CMP

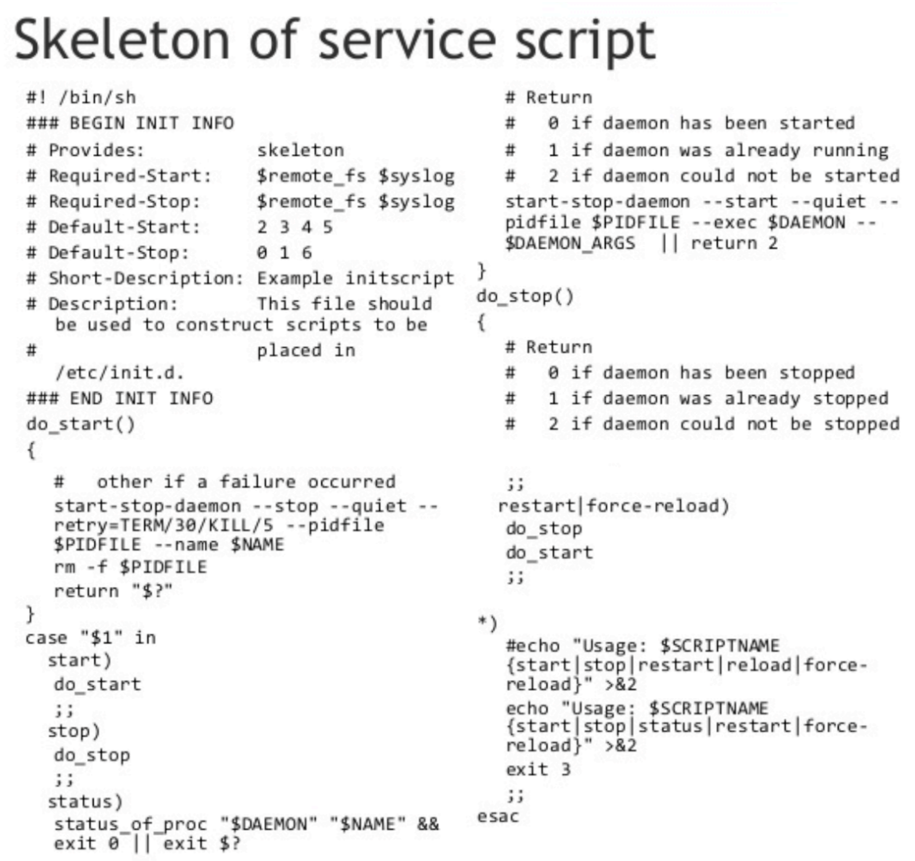




Run Levels on UNIX/Linux like OS

For example,
S12syslog is to start the syslog deamon, which has the sequence number of 12.
S80sendmail is to start the sendmail daemon, which has the sequence number of 80.
K92iptables which has a sequence number of 92 is not start in that particular run level.

Level	Description
0	halt, for power down
1	Single user mode (limited)
2	Multiuser, without NFS
3	Full multiuser mode
4	Unused (User definable)
5	as 3 + X11 – GUI manager
6 S	reboot, for restart system single user (full)

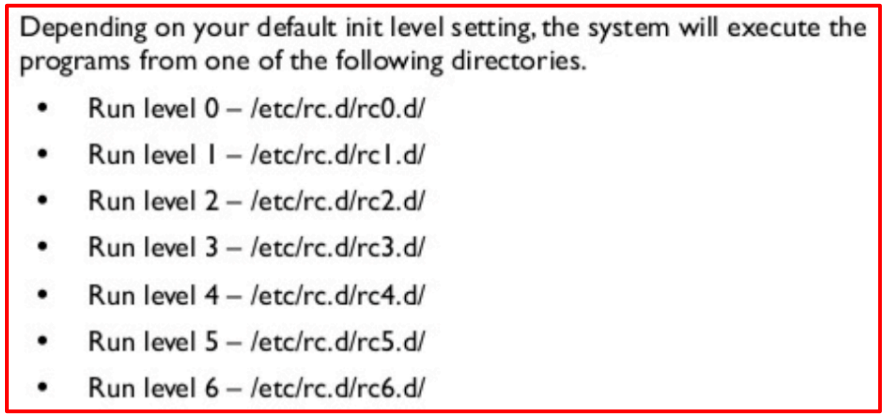


Skeleton of service script

```
#! /bin/sh
### BEGIN INIT INFO
# Provides:          skeleton
# Required-Start:    $remote_fs $syslog
# Required-Stop:     $remote_fs $syslog
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Example initscript
# Description:       This file should
   be used to construct scripts to be
#                    placed in
   /etc/init.d.
### END INIT INFO
do_start()
{

   #   other if a failure occurred
   start-stop-daemon --stop --quiet --
   retry=TERM/30/KILL/5 --pidfile
   $PIDFILE --name $NAME
   rm -f $PIDFILE
   return "$?"
}
case "$1" in
  start)
   do_start
   ;;
  stop)
   do_stop
   ;;
  status)
   status_of_proc "$DAEMON" "$NAME" &&
   exit 0 || exit $?
```

```
   # Return
   #   0 if daemon has been started
   #   1 if daemon was already running
   #   2 if daemon could not be started
   start-stop-daemon --start --quiet --
   pidfile $PIDFILE --exec $DAEMON --
   $DAEMON_ARGS  || return 2
}
do_stop()
{
   # Return
   #   0 if daemon has been stopped
   #   1 if daemon was already stopped
   #   2 if daemon could not be stopped


   ;;
  restart|force-reload)
   do_stop
   do_start
   ;;

*)
   #echo "Usage: $SCRIPTNAME
   {start|stop|restart|reload|force-
   reload}" >&2
   echo "Usage: $SCRIPTNAME
   {start|stop|status|restart|force-
   reload}" >&2
   exit 3
   ;;
esac
```

Task. See content of this directory's for your system:

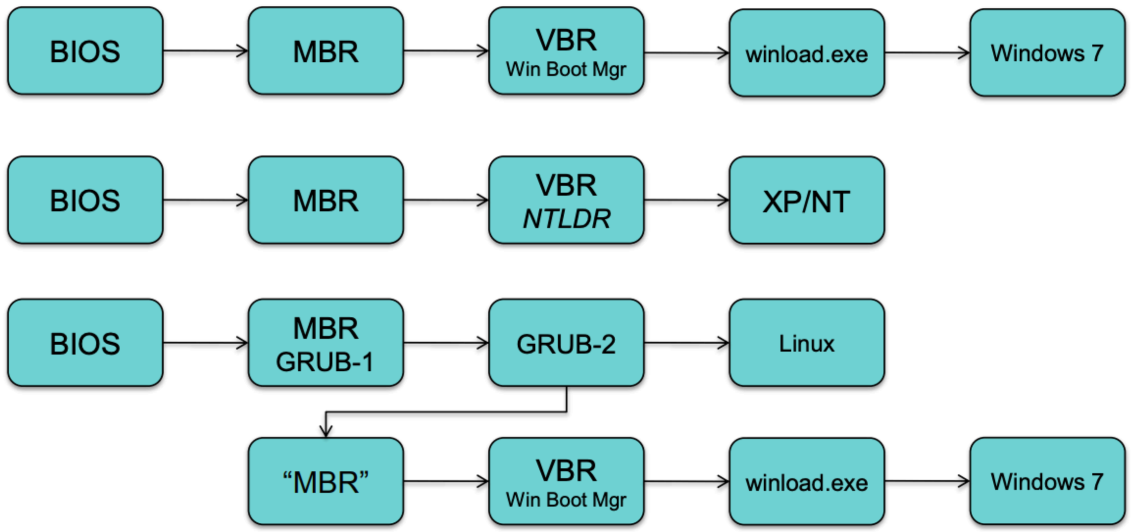


Depending on your default init level setting, the system will execute the programs from one of the following directories.

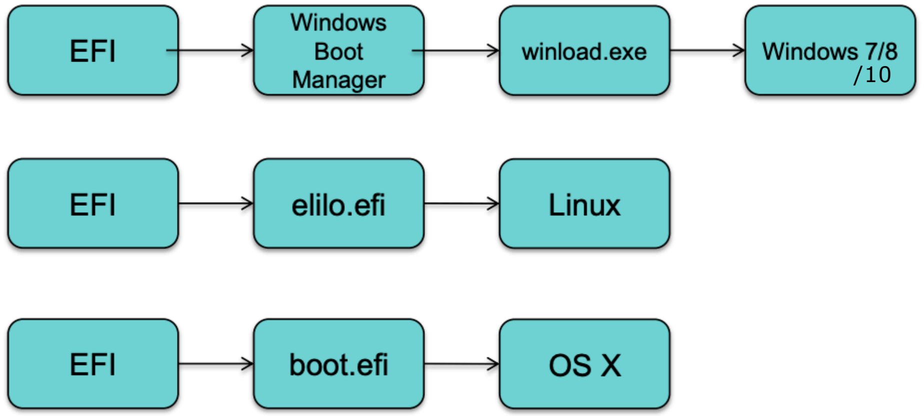
- Run level 0 – /etc/rc.d/rc0.d/
- Run level 1 – /etc/rc.d/rc1.d/
- Run level 2 – /etc/rc.d/rc2.d/
- Run level 3 – /etc/rc.d/rc3.d/
- Run level 4 – /etc/rc.d/rc4.d/
- Run level 5 – /etc/rc.d/rc5.d/
- Run level 6 – /etc/rc.d/rc6.d/

OS Boot Process: Summary

- BIOS/MBR OS Booting



- UEFI/GPT OS Booting



2. Mass-Storage Systems

- Overview of Mass Storage Structure
- Disk Structure
- Disk Attachment
- Disk Management
- Disk Scheduling
- Swap-Space Management
- RAID Structure
- Stable-Storage Implementation

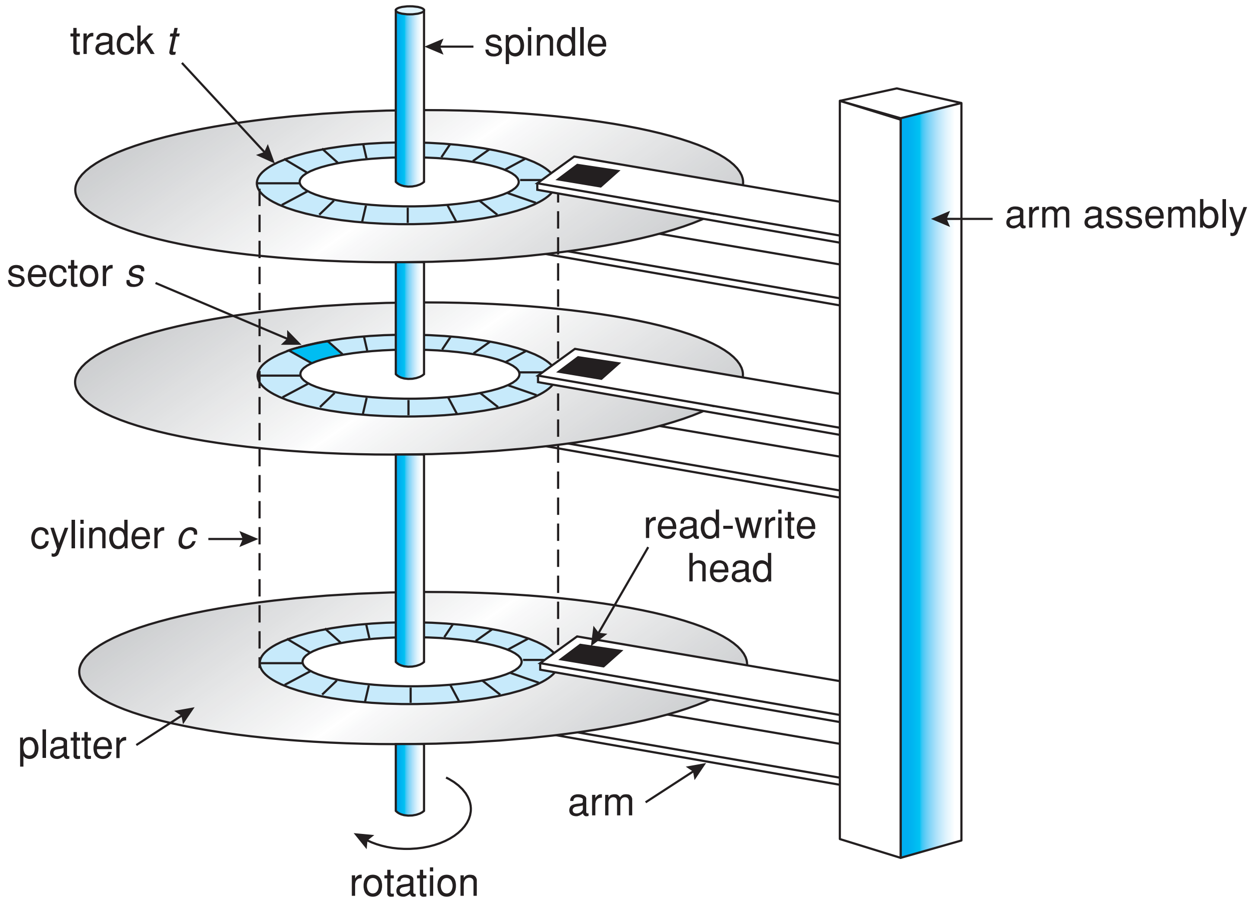
Objectives

- To describe the physical structure of secondary storage devices and its effects on the uses of the devices
- To explain the performance characteristics of mass-storage devices
- To evaluate disk scheduling algorithms
- To discuss operating-system services provided for mass storage, including RAID

Overview of Mass Storage Structure

- **Magnetic disks** provide bulk of secondary storage of modern computers
  - Drives rotate at 60 to 250 times per second
  - **Transfer rate** is rate at which data flow 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 bad
- Disks can be removable
- Drive attached to computer via **I/O bus**
  - Busses vary, including **EIDE**, **ATA**, **SATA**, **USB**, **Fibre Channel**, **SCSI, SAS, Firewire**
  - **Host controller** in computer uses bus to talk to **disk controller** built into drive or storage array

Moving-head Disk Mechanism



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

Magnetic Disks

- Platters range from .85” to 14” (historically)
  - Commonly 5,25’’, 3.5”, 2.5”, 1.8”, 1’’
- 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 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

- Disk Interfaces
  - IDE-EIDE/ATA-ATAPI (Integrated Drive Electronics)
  - SATA, SATA 2, SATA 3 (Serial Advanced Technology Attachment)
  - SCSI (Small Computer System Interface)
  - FC-AL (Fibre Channel- Arbitrated Loop)
  - SAS (Serial Attached SCSI)

RPMs (From Wikipedia)

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

Magnetic Disk Performance

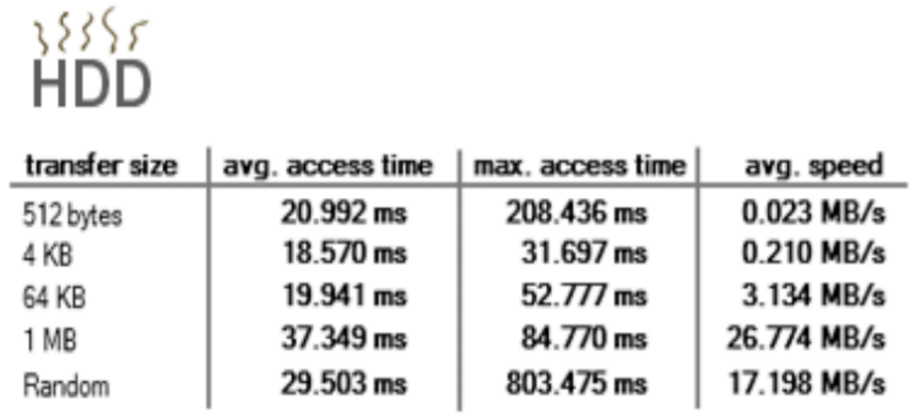
- **Access Latency** = **Average access time** = average seek time + average latency
  - For fastest disk 3ms + 2ms = 5ms
  - For slow disk 9ms + 5.56ms = 14.56ms

- Average I/O time = average access time + (amount to transfer / transfer rate) + controller overhead

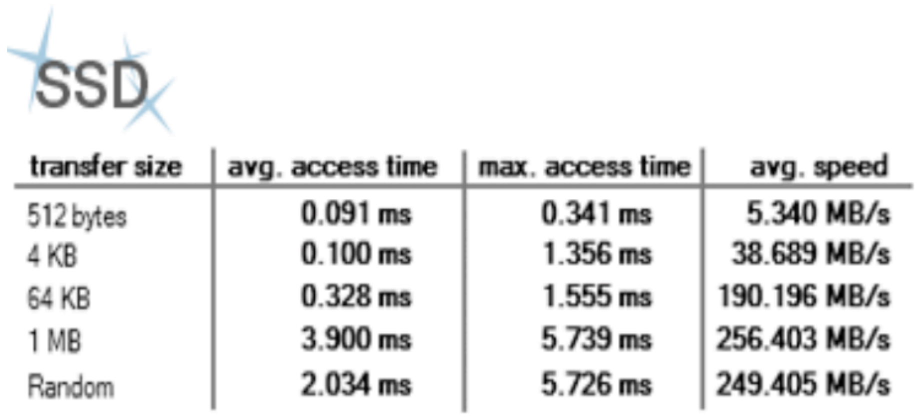
- For example to transfer a 4KB block on a 7200 RPM disk with a 5ms average seek time, 1Gb/sec transfer rate with a .1ms controller overhead =
  - 5ms + 4.17ms + 0.1ms + transfer time =
  - Transfer time = 4KB / 1Gb/s * 8Gb / GB * 1GB / 1024^2KB = 32 / (1024^2) = 0.031 ms
  - Average I/O time for 4KB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



transfer size	avg. access time	max. access time	avg. speed
512 bytes	20.992 ms	208.436 ms	0.023 MB/s
4 KB	18.570 ms	31.697 ms	0.210 MB/s
64 KB	19.941 ms	52.777 ms	3.134 MB/s
1 MB	37.349 ms	84.770 ms	26.774 MB/s
Random	29.503 ms	803.475 ms	17.198 MB/s



transfer size	avg. access time	max. access time	avg. speed
512 bytes	0.091 ms	0.341 ms	5.340 MB/s
4 KB	0.100 ms	1.356 ms	38.689 MB/s
64 KB	0.328 ms	1.555 ms	190.196 MB/s
1 MB	3.900 ms	5.739 ms	256.403 MB/s
Random	2.034 ms	5.726 ms	249.405 MB/s

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 Structure

- 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

Disk Attachment

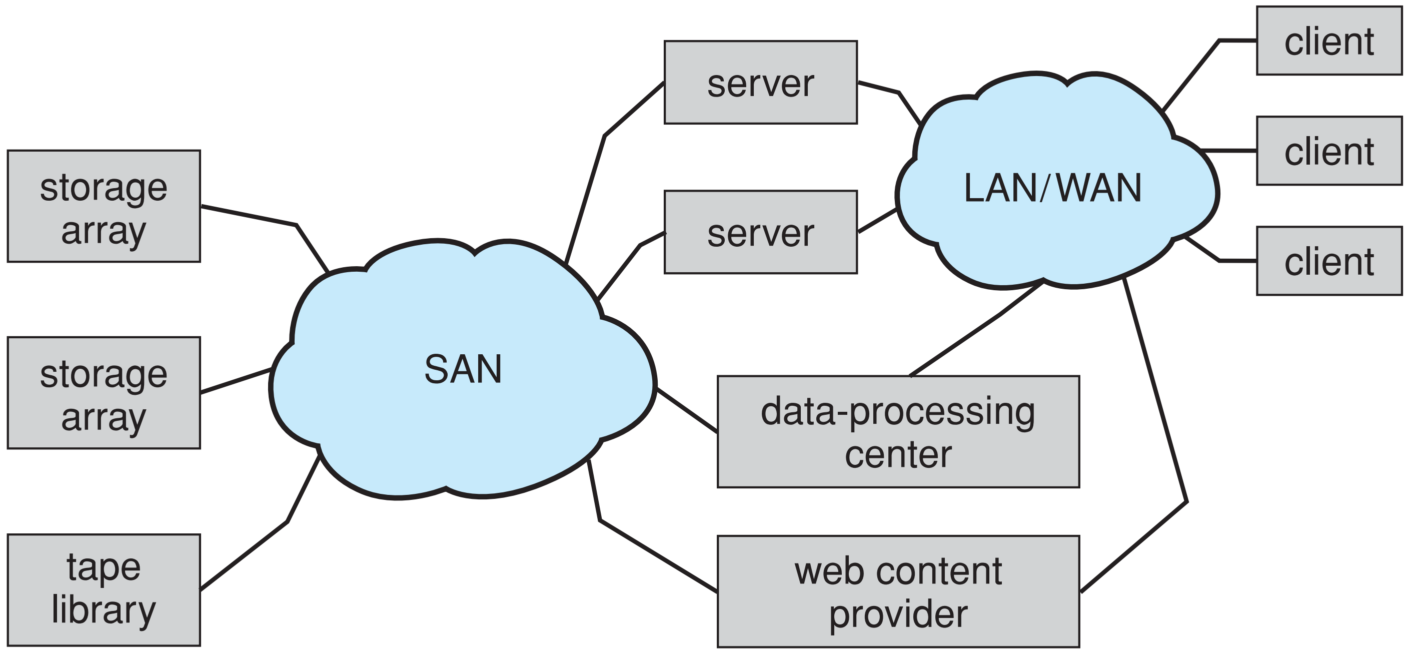
- Host-attached storage accessed through I/O ports talking to I/O busses
- SCSI itself is a bus, up to 16 devices on one cable, **SCSI initiator** requests operation and **SCSI targets** perform tasks
  - Each target can have up to 8 **logical units** (disks attached to device controller)
- FC is high-speed serial architecture
  - Can be switched fabric with 24-bit address space – the basis of **storage area networks (SANs)** in which many hosts attach to many storage units
- I/O directed to bus ID, device ID, logical unit (LUN)

Storage Array

- Can just attach disks, or arrays of disks
- 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

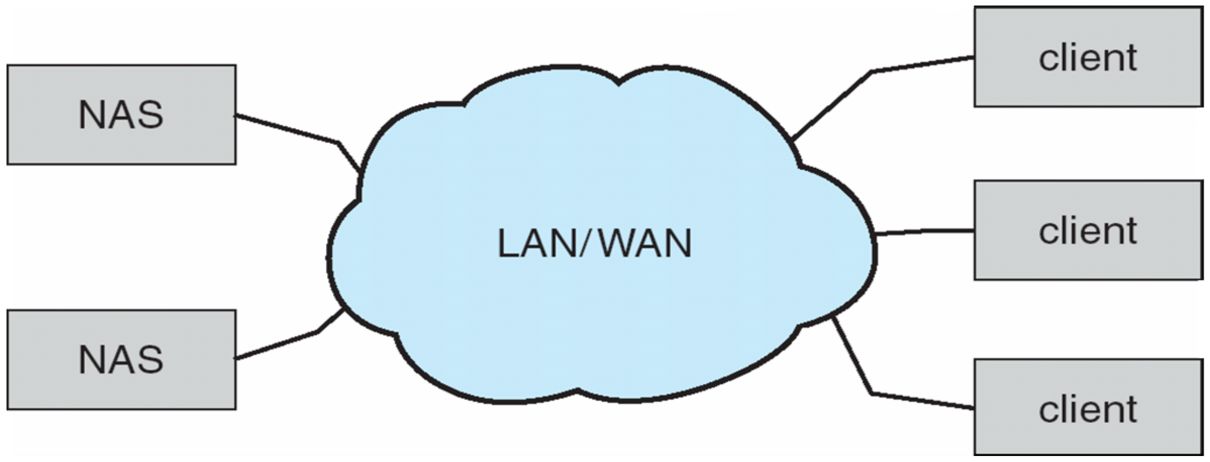


Storage Area Network (Cont.)

- SAN is one or more storage arrays
  - Connected to one or more Fibre Channel switches
- 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
  - Over low-latency Fibre Channel fabric
- Why have separate storage networks and communications networks?
  - Consider iSCSI, FCOE

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)



Disk Management

- **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
- 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
- Raw disk access for apps that want to do their own block management, keep OS out of the way (databases for example)
- Boot block initializes system
  - The bootstrap is stored in ROM
  - **Bootstrap loader** program stored in boot blocks of boot partition
- Methods such as **sector sparing** used to handle bad blocks

Disk Scheduling

- The operating system is responsible for using hardware efficiently — for the disk drives, this means having a fast access time and disk bandwidth
- 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

Disk Scheduling (Cont.)

- There are many sources of disk I/O request
  - OS
  - System processes
  - 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
- Note that drive controllers have small buffers and can manage a queue of I/O requests (of varying “depth”)

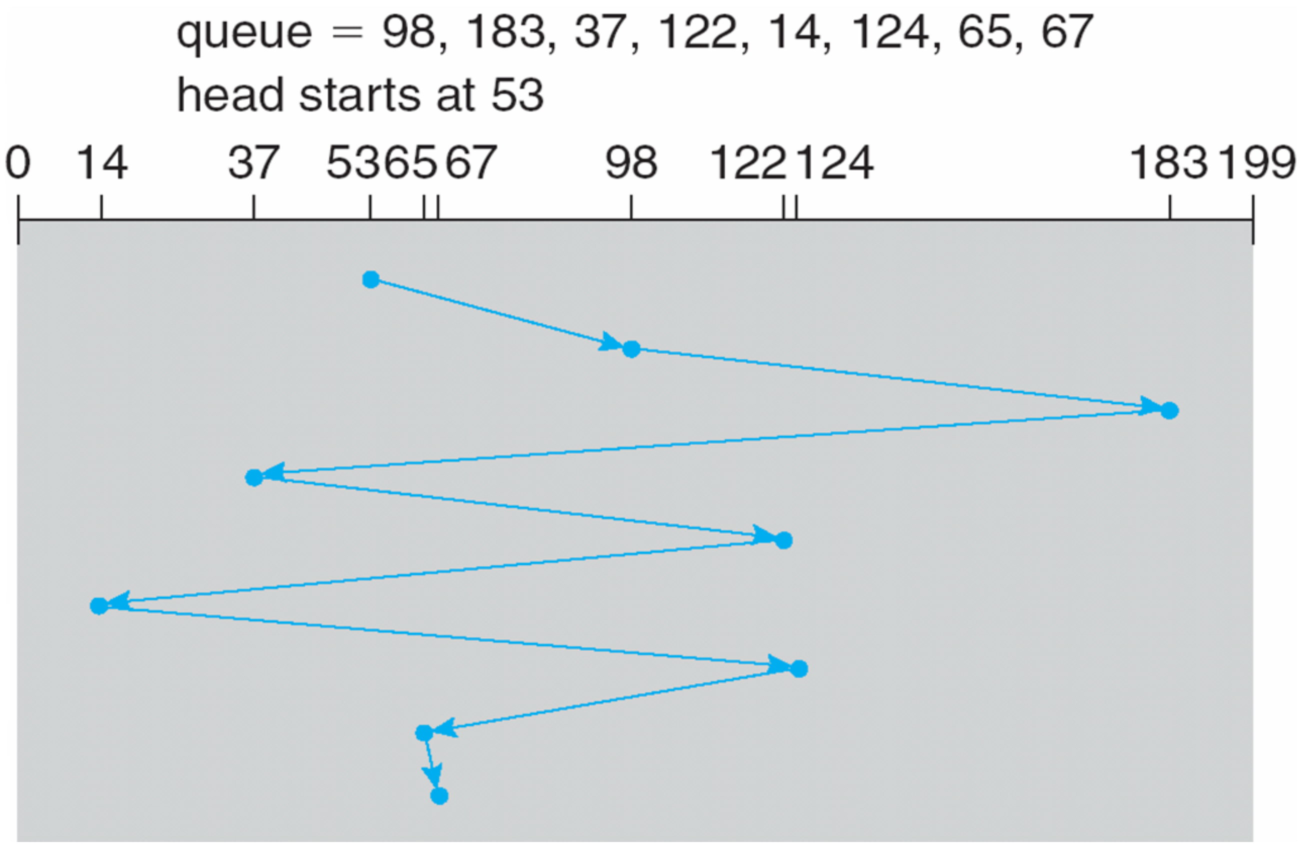
- Several algorithms exist to schedule the servicing of disk I/O requests
- The analysis is true for one or many platters
- We illustrate scheduling algorithms with a request queue (0-199)

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

Head pointer 53

FCFS

- First Come First Served
- Illustration shows total head movement of 640 cylinders

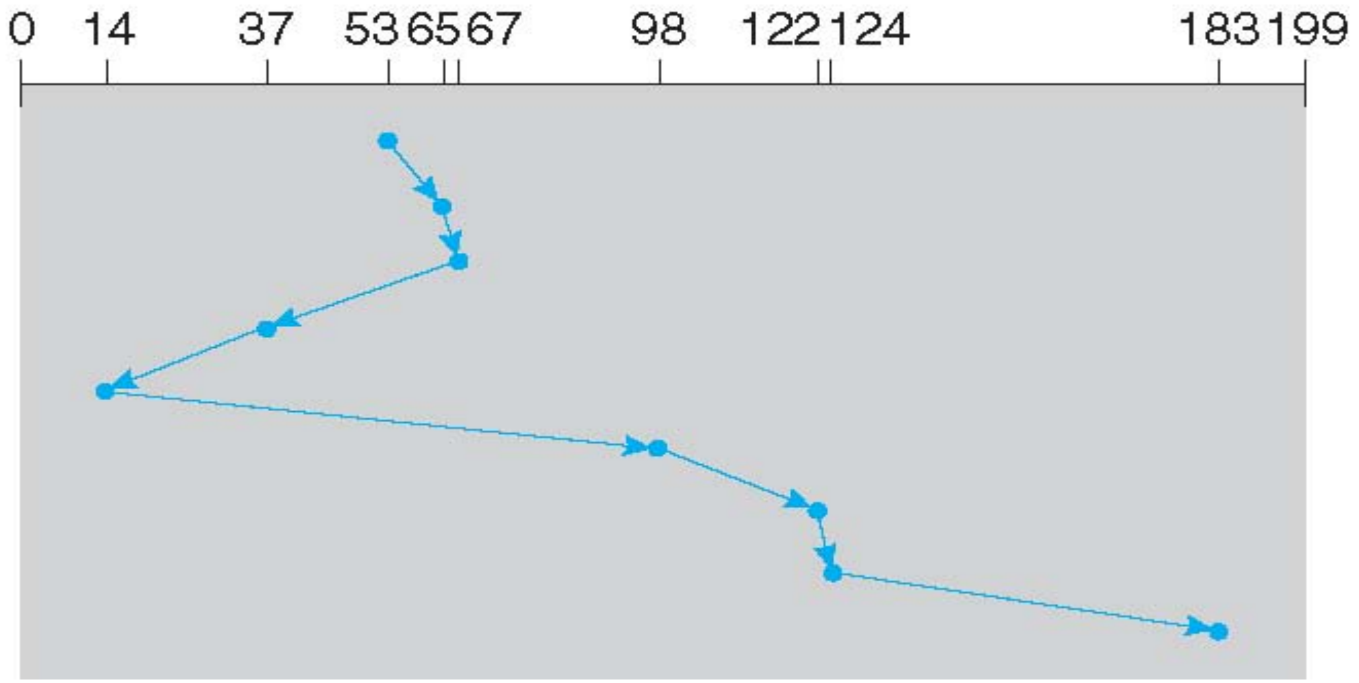


SSTF

- Shortest Seek Time First selects the request with the minimum seek time from the current head position
- SSTF scheduling is a form of SJF scheduling; may cause starvation of some requests
- Illustration shows total head movement of 236 cylinders

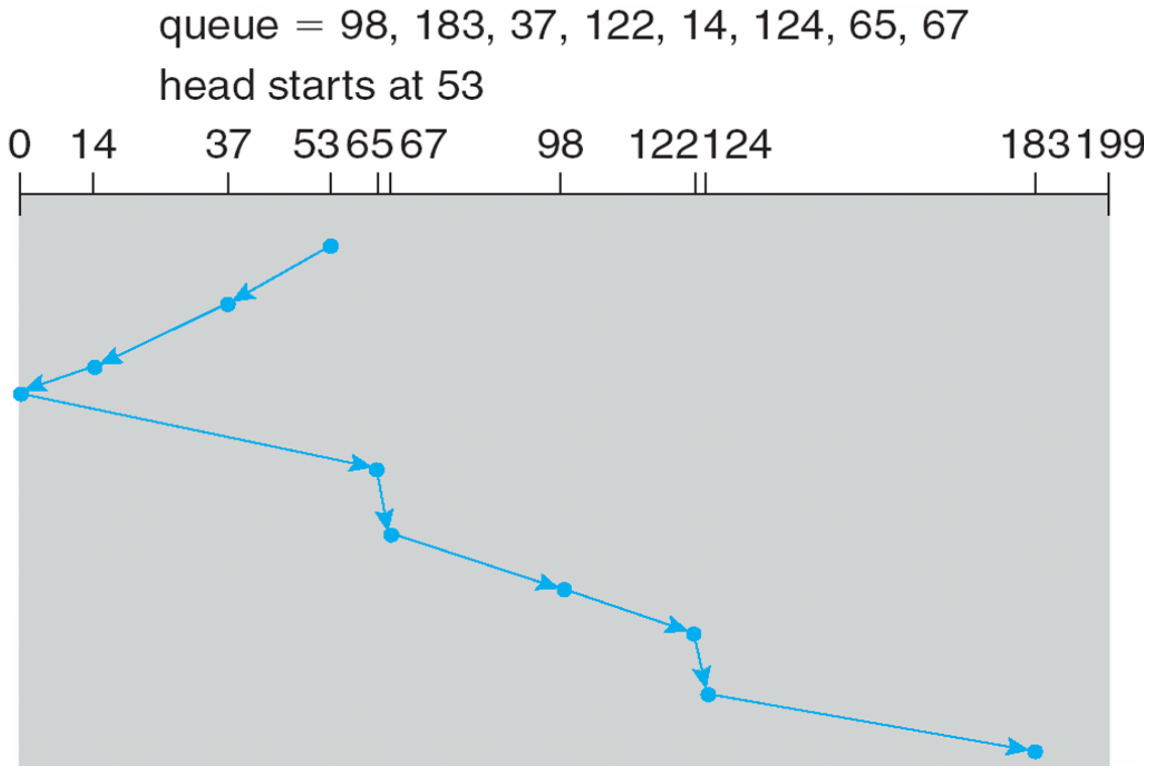
queue = 98, 183, 37, 122, 14, 124, 65, 67

head starts at 53



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

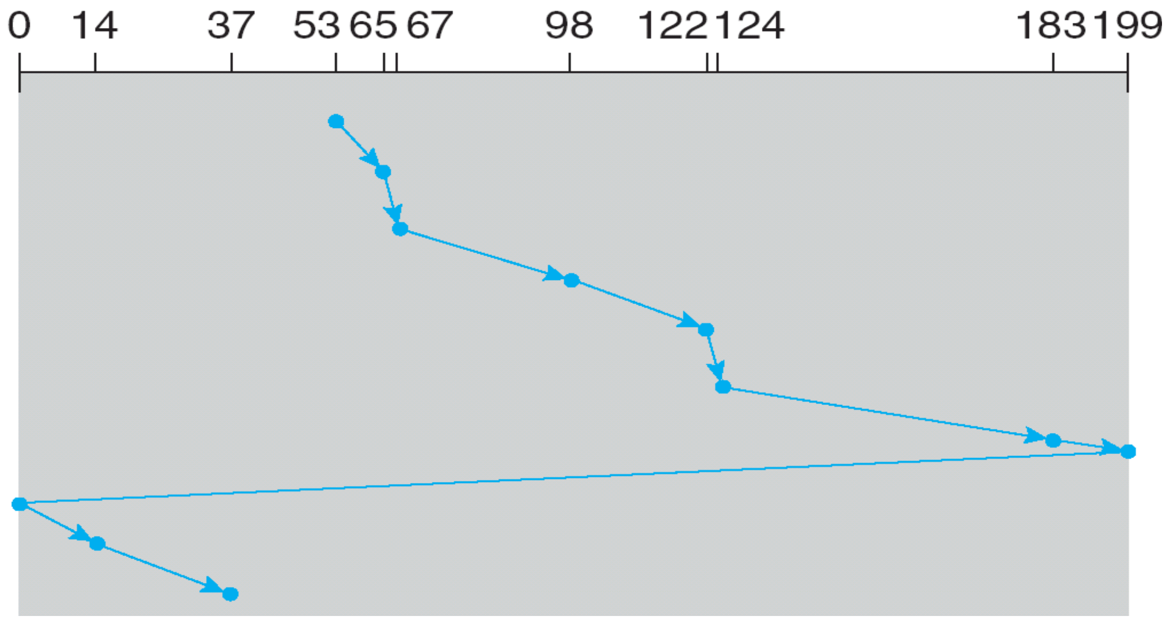


C-SCAN

- 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
- Total number of cylinders may be less or more then on SCAN.

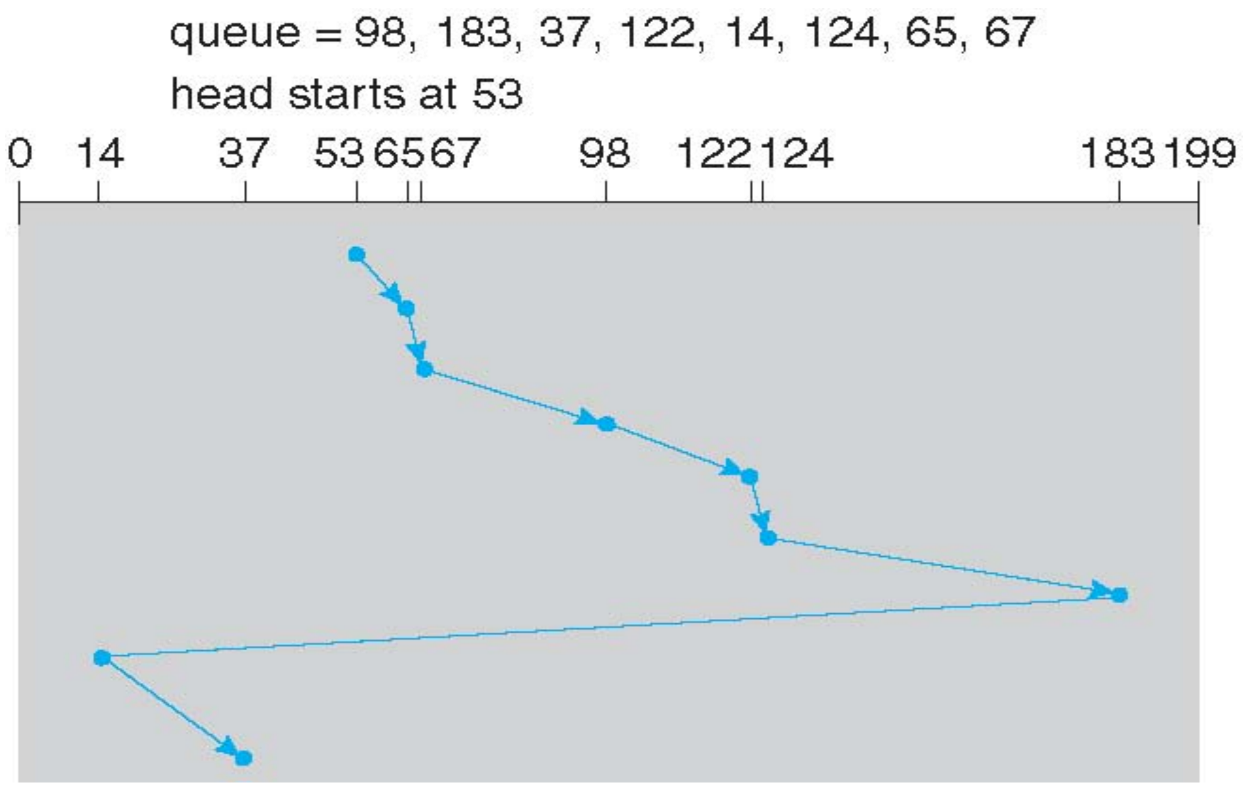
queue = 98, 183, 37, 122, 14, 124, 65, 67

head starts at 53



C-LOOK

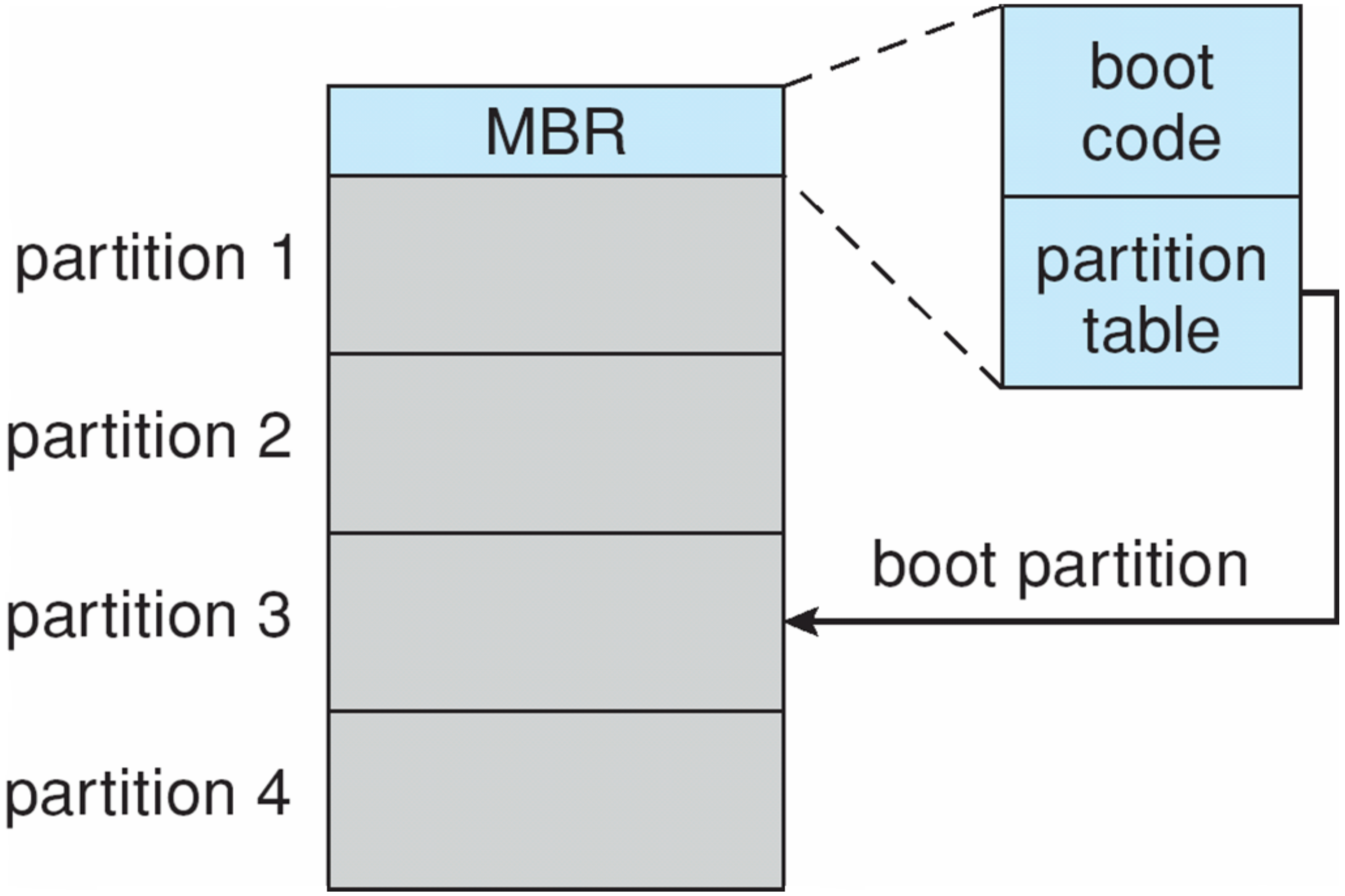
- LOOK a version of SCAN, C-LOOK a version of C-SCAN
- Arm only goes as far as the last request in each direction, then reverses direction immediately, without first going all the way to the end of the disk
- Total number of cylinders less then on C-SCAN?



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
  - Less starvation
- Performance depends on the number and types of requests
- Requests for disk service can be influenced by the file-allocation method
  - And metadata layout
- 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 algorithm
- What about rotational latency?
  - Difficult for OS to calculate
- How does disk-based queueing effect OS queue ordering efforts?

Booting from a Disk in Windows

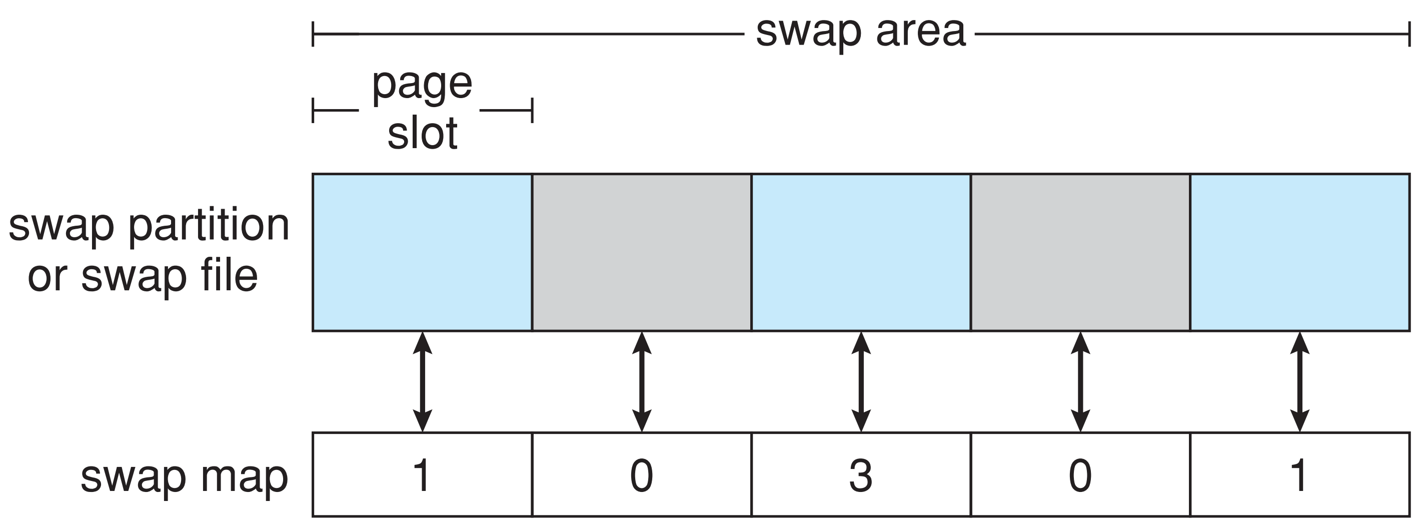


Swap-Space Management

What is Swap Space?

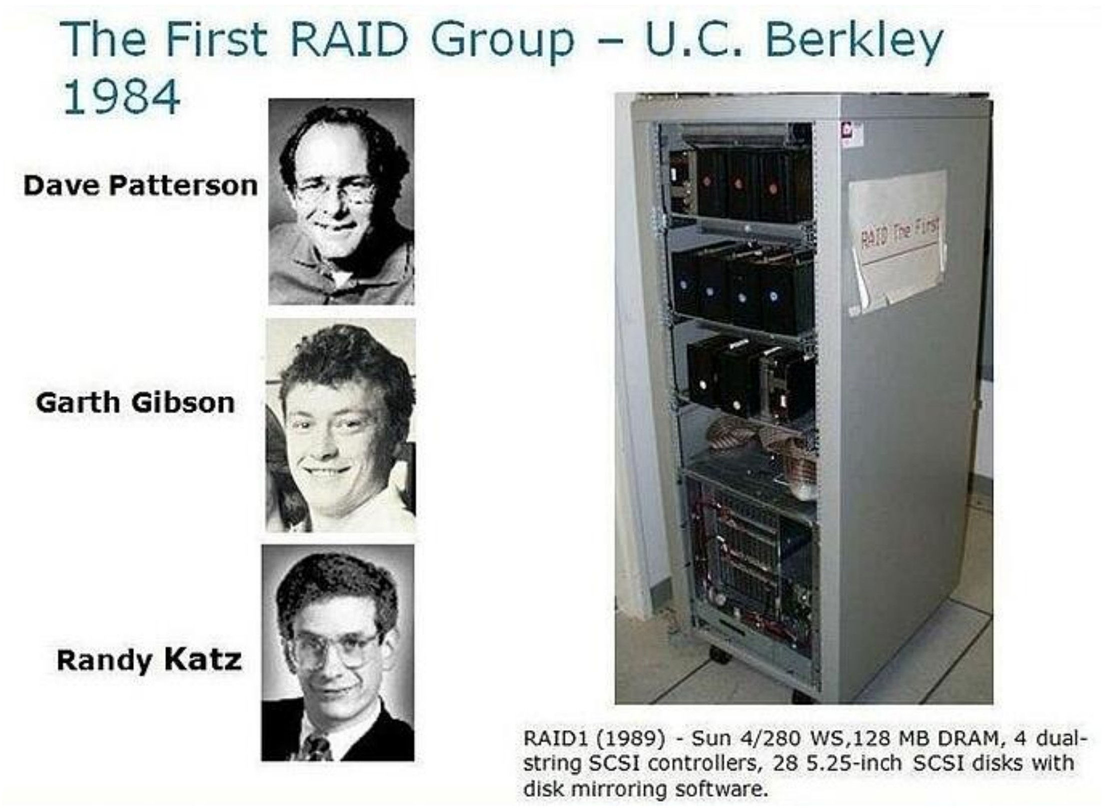
- Swap-space — Virtual memory uses disk space as an extension of main memory
- If RAM is full, inactive pages in memory are moved to the swap space.
- Swap space is located on hard drives, which have a slower access time than physical memory.
- Swap space can be a dedicated swap partition (recommended), a swap file, or a combination of swap partitions and swap files.
- The size of your swap
  - Swap=2*RAM for physical RAM < 2 GB.
  - Swap=2 GB for physical RAM > 2 GB.
  - Swap should never be less than 32 MB.
- Swap-space management
  - 4.3BSD allocates swap space when process starts; holds text segment (the program) and data segment
  - Kernel uses **swap maps** to track swap-space use
  - Solaris 2 allocates swap space only when a dirty page is forced out of physical memory, not when the virtual memory page is first created
    - File data written to swap space until write to file system requested
    - Other dirty pages go to swap space due to no other home
    - Text segment pages thrown out and reread from the file system as needed

Data Structures for Swapping on Linux Systems



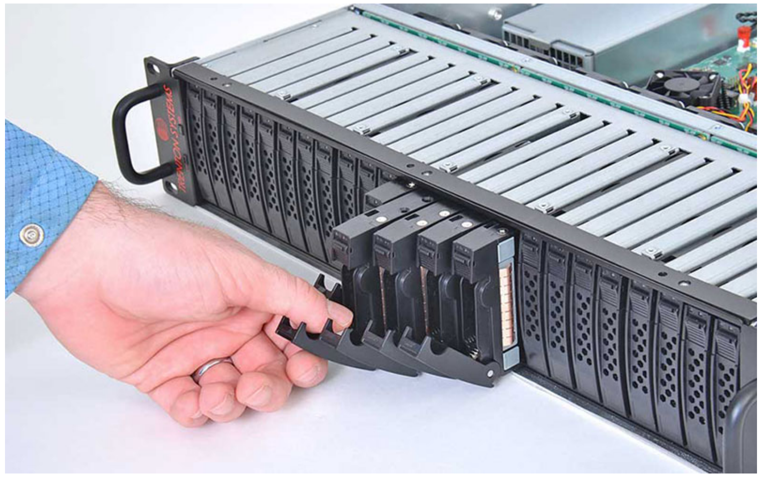
RAID Structure

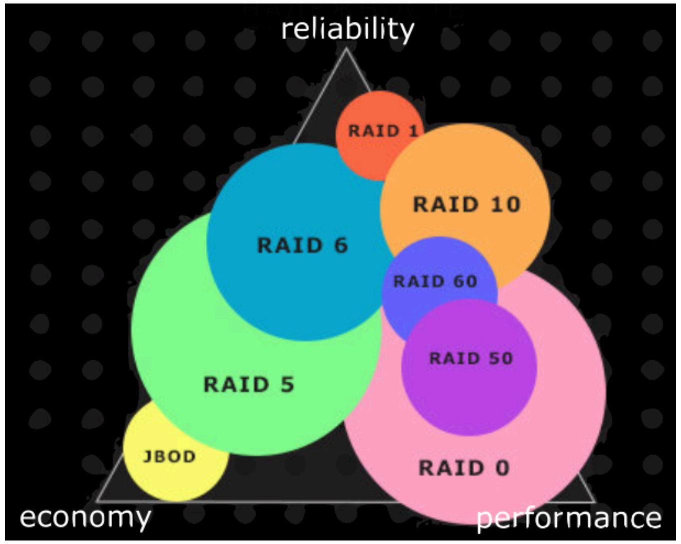
- RAID stands for Redundant Array of Inexpensive (Independent) Disks
  - multiple disk drives provides reliability via **redundancy**
- Increases the **mean time to failure**
- **Mean time to repair –** exposure time when another failure could cause data loss
- **Mean time to data loss** based on above factors
- If mirrored disks fail independently, consider disk with 1300,000 mean time to failure and 10 hour mean time to repair
  - Mean time to data loss is $100,000^2 / (2 * 10) = 500 * 10^6$ hours, or 57,000 years!
- Frequently combined with **NVRAM** to improve write performance
- RAID is arranged into many different levels


The First RAID Group – U.C. Berkley 1984
Dave Patterson
Garth Gibson
Randy Katz
RAID The First
RAID1 (1989) - Sun 4/280 WS,128 MB DRAM, 4 dual-string SCSI controllers, 28 5.25-inch SCSI disks with disk mirroring software.

RAID Structure (Cont.)

- Several improvements in disk-use techniques involve the use of multiple disks working cooperatively
- **JBOD** stands for "Just a Bunch Of Disks" inside of a single storage enclosure.
- Disk **striping** (**RAID 0**) uses a group of disks as one storage unit
- 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,** also known as RAID 10) 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





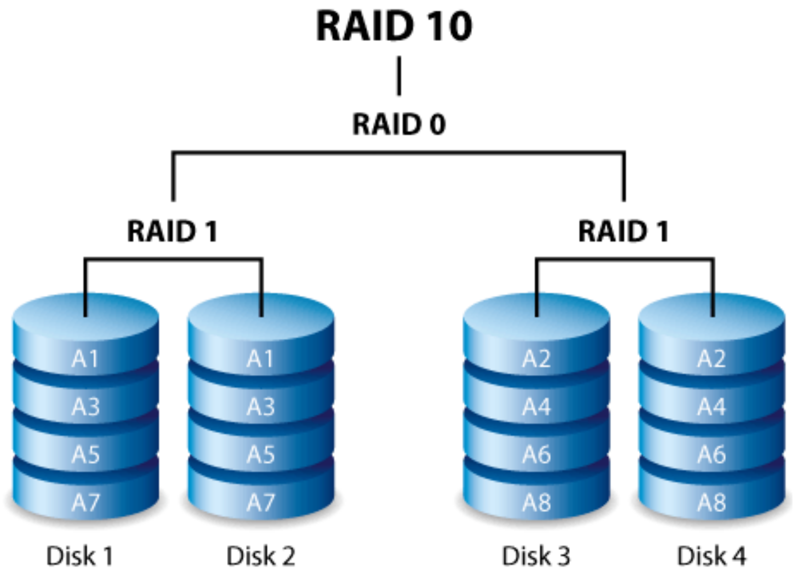
RAID Levels

Who Should Use RAID?

Those who need to keep large quantities of data.

Primary reasons to use RAID:

- Enhanced speed
- Increased storage capacity using a single virtual disk
- Lessened impact of a disk failure



There are two possible RAID approaches:

Hardware RAID

- The hardware-based array manages the RAID subsystem independently from the host and presents to the host only a single disk per RAID array.
- An example of a Hardware RAID device would be one that connects to a SCSI controller and presents the RAID arrays as a single SCSI drive. All RAID handling "intelligence" moves into external disk subsystem.

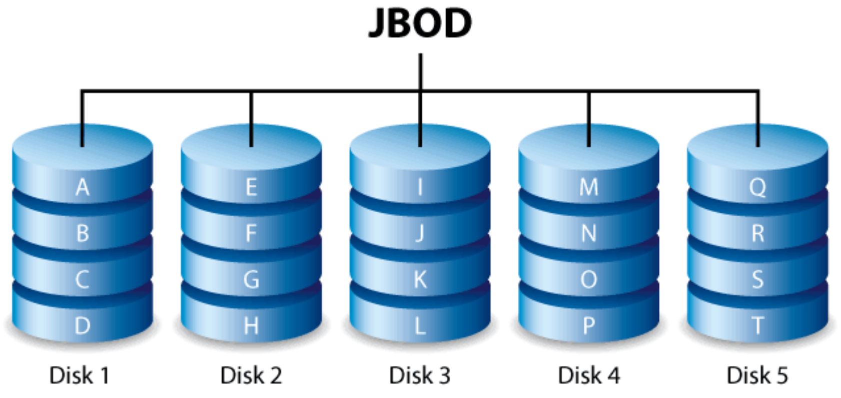
Software RAID

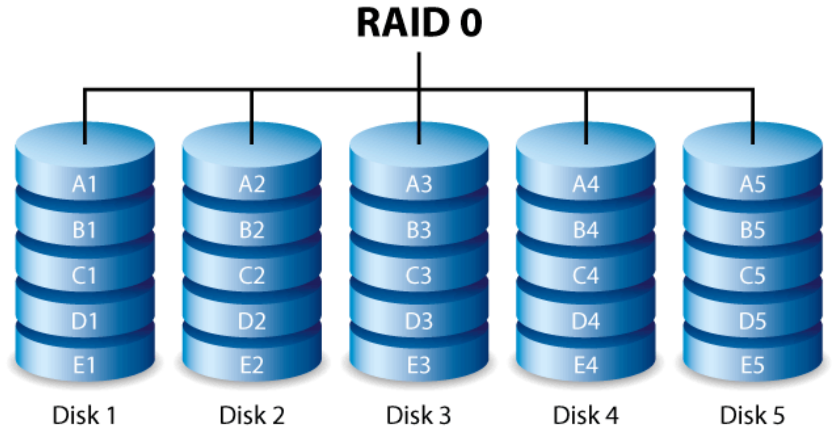
- Software RAID implements the various RAID levels in the kernel disk (block device) code. It offers the cheapest possible solution, as expensive disk controller cards or hot-swap chassis are not required. Software RAID also works with cheaper IDE disks as well as SCSI disks.
- With today's fast CPUs, Software RAID performance can excel against Hardware RAID.

RAID Levels

In all the diagrams mentioned below:

- A, B, C, D, E, F … – represents blocks
- b1, b2, b3 … – represents bits





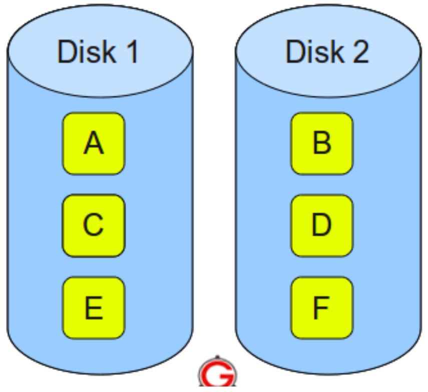
JBOD vs RAID 0

JBOD, which stands for Just a Bunch of Disks or Just a Bunch of Drives, is a storage architecture consisting of numerous disk drives inside of a single storage enclosure.

- Minimum 2 disks
- Increased storage capacity using a single virtual disk
- Disks in a JBOD configuration store data sequentially. For example, data is written to Disk 1 first. Once Disk 1 is full, data will be written to Disk 2, then Disk 3, etc.
- Two advantages to this level of RAID are the availability of 100% of the disks' total storage capacity and easy expansion.
- if a drive in a **JBOD** set dies then it may be easier to recover the files on the other drives
- Capacity usage = n*C
  (where N is the number of disks in the array, and C is their capacity)

<u>Disadvantage:</u> No performance, no redundancy

RAID Levels

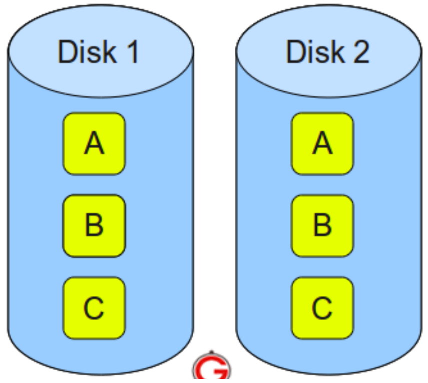


RAID0

Following are the key points to remember for RAID level 0.

- Block Striped.
- Minimum 2 disks.
- Excellent read and write performance (as blocks are striped).
- Don't use this for any critical system.
- Capacity usage = n*C

Disadvantage: No redundancy (no mirror, no parity).

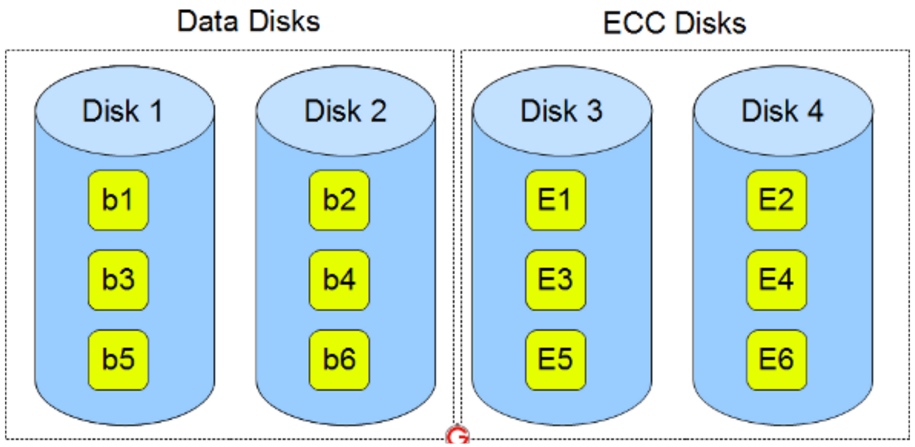


RAID1

- Block Mirrored.
- Minimum 2 disks.
- Excellent read performance (as blocks are striped).
- Good write performance (no striping, no parity).
- Excellent redundancy (as blocks are mirrored).
- Capacity usage = n*C/2

Disadvantage: for 1 logical space need 2 and more physical space

RAID Levels

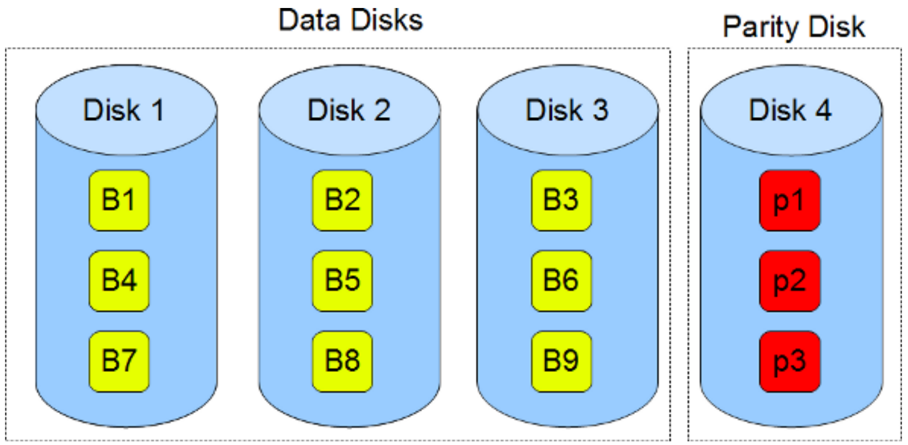


RAID2

- Bits Striped and stores ECC (error correction codes).
- Minimum 4 disks. You need two groups of disks. One group of disks are used to write the data, another group is used to write the **Hamming ECC**
- In the diagram b1, b2, b3 are bits. E1, E2, E3 are ECC.
- When data is written to the disks, it calculates the ECC code for the data on the fly, and stripes the data bits to the data-disks, and writes the ECC code to the redundancy disks.
- When data is read from the disks, it also reads the corresponding ECC code from the redundancy disks, and checks whether the data is consistent. If required, it makes appropriate corrections on the fly.
- This uses lot of disks and can be configured in different disk configuration. Some valid configurations are 1) 10 disks for data and 4 disks for ECC 2) 4 disks for data and 3 disks for ECC

Disadvantage: This is expensive and implementing it in a RAID controller is complex, and ECC is redundant now-a-days, as the hard disk themselves can do this.

RAID2 is not used anymore.



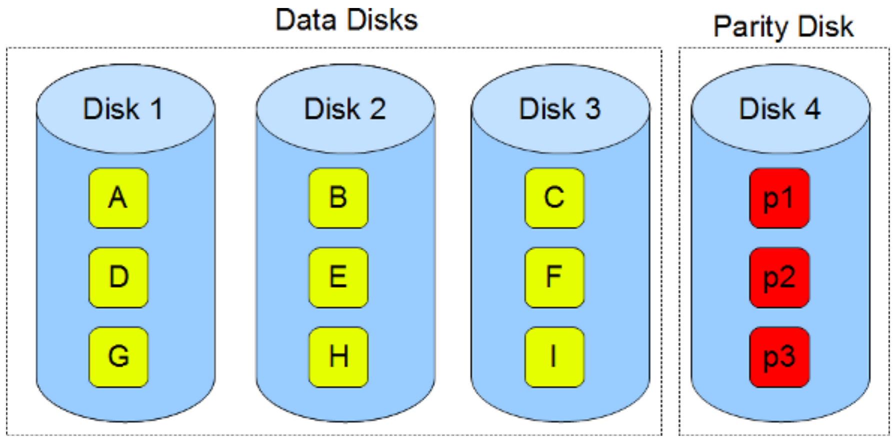
RAID3

- Bytes Striped and Dedicated Parity Disk.
- Minimum 3 disks.
- In the diagram B1, B2, B3 are bytes. p1, p2, p3 are parities.
- Uses multiple data disks, and a dedicated disk to store parity.
- The disks have to spin in sync to get to the data.
- Sequential read and write will have good performance.
- Capacity usage = (n-1)*C

Disadvantage: Random read / write will have worst performance.

RAID3 is not commonly used.

RAID Levels

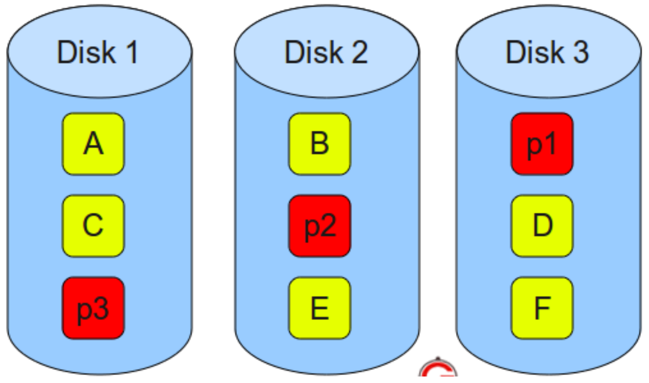


RAID4

- Blocks Striped and Dedicated Parity Disk.
- Minimum 3 disks (2 disks for data and 1 for parity).
- Uses multiple data disks, and a dedicated disk to store parity.
- In the diagram B1, B2, B3 are blocks. p1, p2, p3 are parities.
- Good random reads, as the data blocks are striped.
- Capacity usage = (n-1)*C

Disadvantage: Bad random writes (write to the single parity disk). It is somewhat similar to RAID 3 and 5, but a little different.

RAID4 is not commonly used.

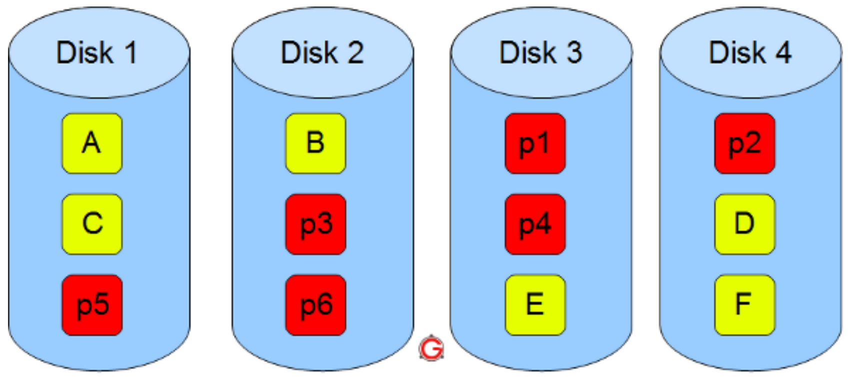


RAID5

- Block Striped and Distributed Parity.
- Minimum 3 disks.
- Good read performance (as blocks are striped).
- Good redundancy (distributed parity).
- Best cost effective option providing both performance and redundancy.
- Capacity usage = (n-1)*C

Disadvantage: Write operations will be slow. Use this for DB that is heavily read oriented.

RAID Levels



RAID6

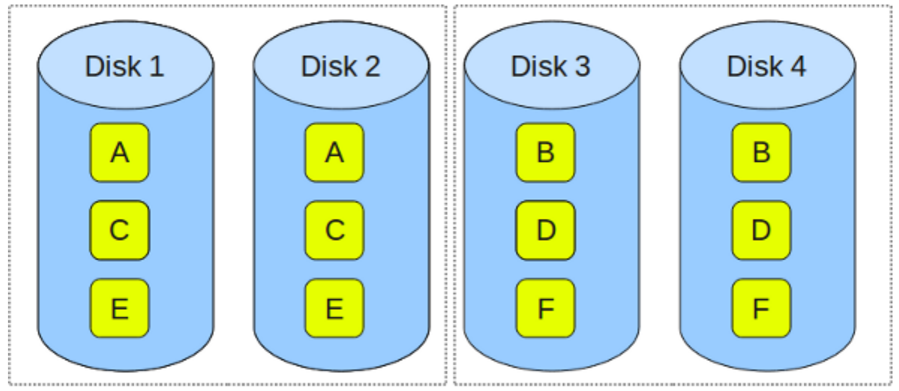
- Block Striped and Two Distributed Parity.
- Minimum 4 disks.
- In the diagram A,B,C,… are blocks. p1,p2,p3,… are parities.
- This creates two parity blocks P and Q for each data block.

  $P = D0 \oplus D1 \oplus D2 \oplus \ldots \oplus Dn-1$

  $Q=(A0 \otimes D0) \oplus (A1 \otimes D1) \oplus (A2 \otimes D2) \oplus \ldots \oplus (An-1 \otimes Dn-1)$

  Ai - is a special coefficient.
- Can handle two disk failure
- Capacity usage = (n-2)*C

Disadvantage: This RAID configuration is complex to implement in a RAID controller, as it has to calculate two parity data for each data block. Write operations will be slow.

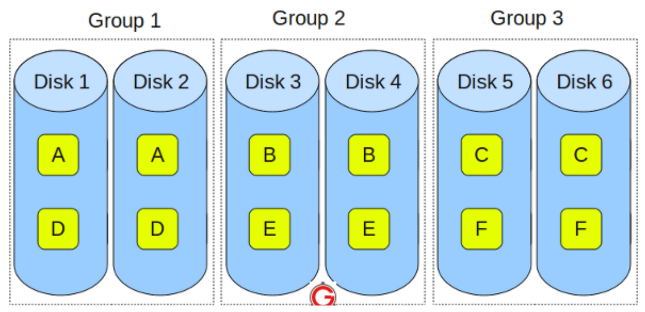


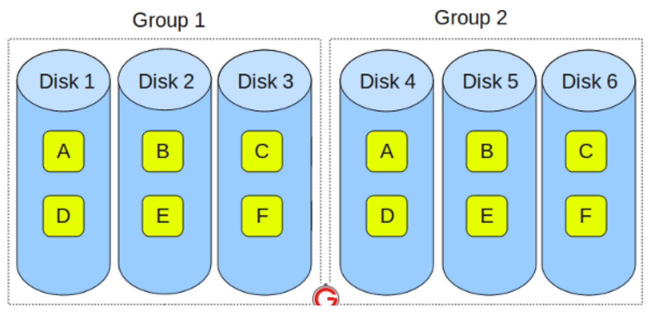
RAID 10

- Block Mirrored and Block Striped, called as “stripe of mirrors”
- Minimum 4 disks.
- Excellent performance (as blocks are striped)
- Excellent redundancy (as blocks are mirrored)
- If you can afford the dollar, this is the BEST option for any mission critical applications (especially databases).
- Capacity usage = n*C/2

Disadvantage: for 1 logical space need 2 and more physical space

RAID 10 (1+0) vs RAID 01 (0+1)





RAID10 (also called as RAID 1+0, or "stripe of mirrors")

- Block Mirrored and Block Striped
- It requires minimum of 4 disks
- Excellent performance (as blocks are striped) and redundancy (as blocks are mirrored)
- To understand this better, group the disks in pair of two (for mirror). For example, if you have a total of 6 disks in RAID 10, there will be three groups–Group 1, Group 2, Group 3 as shown in the above diagram.

RAID01 (also called as RAID 0+1, or "mirror of stripes")

- Is Block Striped and Block Mirrored
- It requires min of 3 disks, but in most cases this will be implemented as min of 4 disks.
- Excellent performance (as blocks are striped) and redundancy (as blocks are mirrored)
- To understand this better, create two groups. For example, if you have total of 6 disks, create two groups with 3 disks each as shown below. In the above example, Group 1 has 3 disks and Group 2 has 3 disks.

In the diagrams A, B, C, D, E and F represents blocks. Main difference between RAID 10 vs RAID 01

- Performance and storage capacity on both RAID 10 and RAID 01 will be the same.
- The main difference is the fault tolerance level. On most implememntations of RAID controllers, RAID 01 fault tolerance is less.
- On RAID 01, since we have only two groups of RAID 0, if two drives (one in each group) fails, the entire RAID 01 will fail.
    - For example, if Disk 1 and Disk 4 fails, both the groups will be down. So, the whole RAID 01 will fail.
- RAID 10 fault tolerance is more, since there are many groups, even if three disks fails (one in each group), the RAID 10 is still functional.
    - In the above RAID 10 example, even if Disk 1, Disk 3, Disk 5 fails, the RAID 10 will still be functional.
- So, given a choice between RAID 10 and RAID 01, always choose RAID 10.

Other RAID 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)
- 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

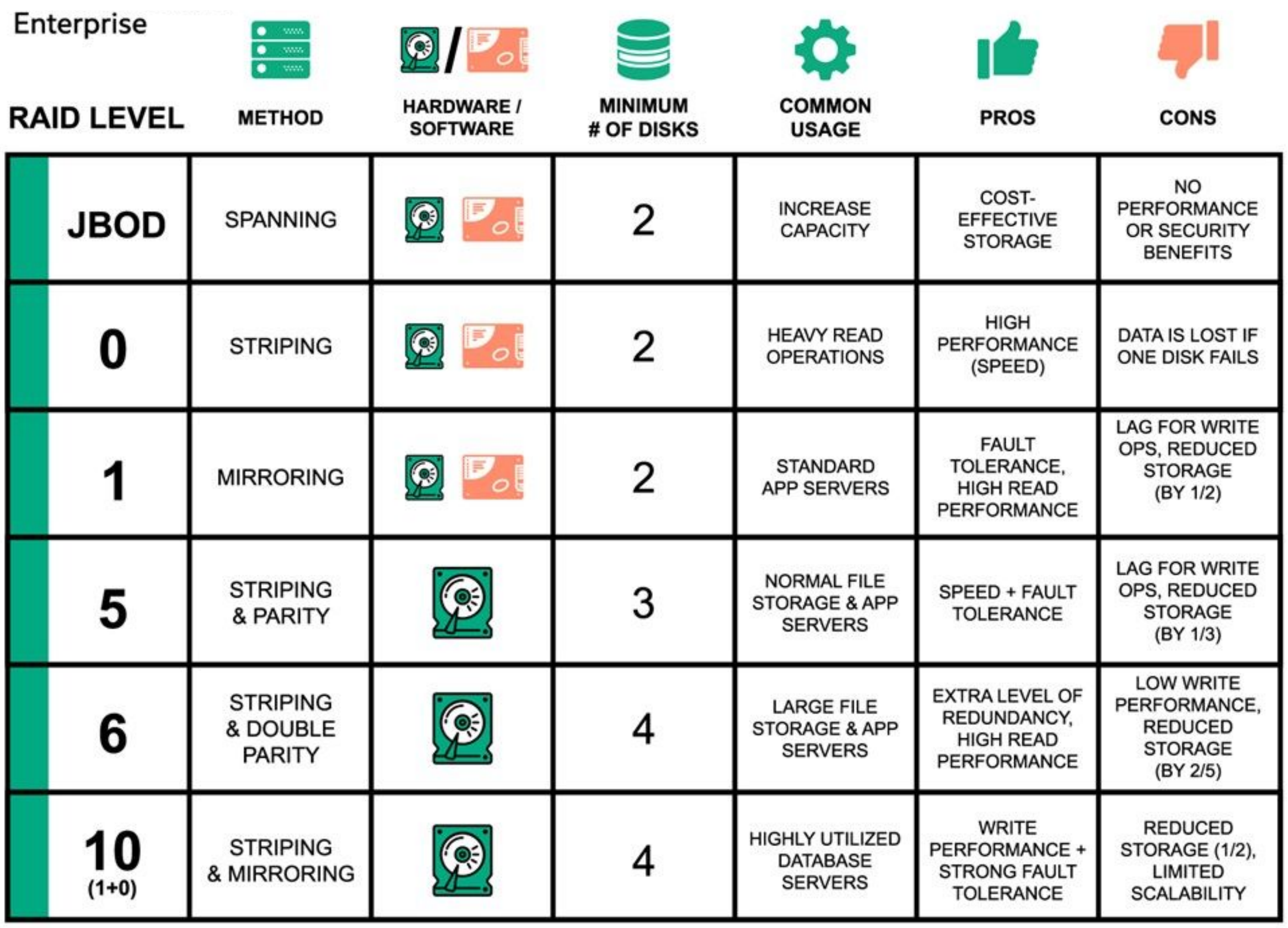
Which RAID is best?

The best RAID configuration for your storage system will depend on whether you value speed, data redundancy or both

Determine your RAID goals by reviewing the following scenarios:

1. If you value speed most of all?
   - Choose RAID 0.
2. If you value data redundancy most of all?
   - That the following drive configurations are fault-tolerant: RAID 1, RAID 5, RAID 6 and RAID 10.
3. Are you a large business or organization with multiple servers and numerous employees who need consistent access to the data stored on those servers?
   - Choose RAID 5, RAID 6 or RAID 10, and go with a hardware RAID controller.
4. Are you a small business or organization where speed isn't as much of a priority as proper record-keeping?
   - Choose RAID 1 or RAID 5, and choose your operating system's software RAID driver.
5. Are you a mission-critical business or organization where a loss of sensitive, classified or other vitally important data could result in headache, financial ruin, serious injury or even death?
   - Choose RAID 6 or RAID 10, and choose a hardware RAID controller.
6. Are you a gamer, photographer, videographer, music producer or other user who values speed and efficiency over fault tolerance?
   - Choose RAID 0 with your operating system's software RAID driver. But be sure to conduct regular backups.

Which RAID is best?

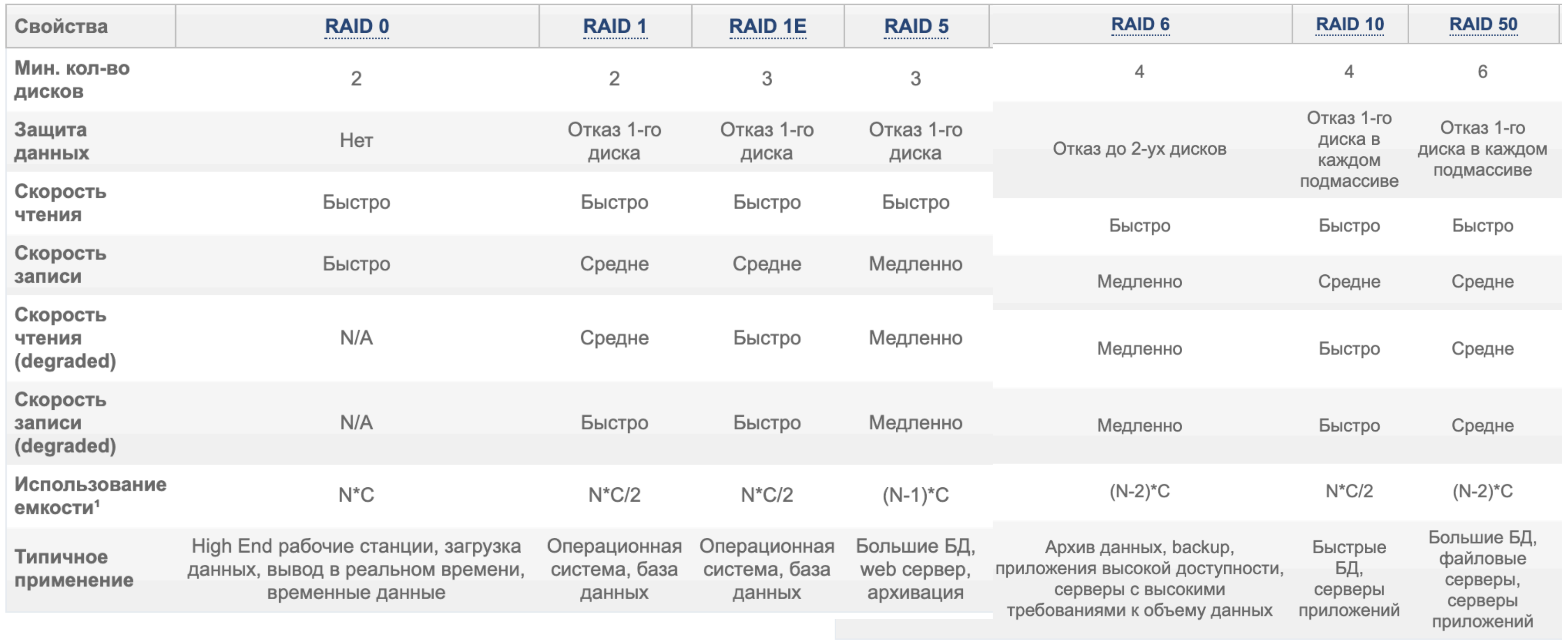


Enterprise

RAID LEVEL	METHOD	HARDWARE / SOFTWARE	MINIMUM # OF DISKS	COMMON USAGE	PROS	CONS
JBOD	SPANNING	🟩 🟧	2	INCREASE CAPACITY	COST-EFFECTIVE STORAGE	NO PERFORMANCE OR SECURITY BENEFITS
0	STRIPING	🟩 🟧	2	HEAVY READ OPERATIONS	HIGH PERFORMANCE (SPEED)	DATA IS LOST IF ONE DISK FAILS
1	MIRRORING	🟩 🟧	2	STANDARD APP SERVERS	FAULT TOLERANCE, HIGH READ PERFORMANCE	LAG FOR WRITE OPS, REDUCED STORAGE (BY 1/2)
5	STRIPING & PARITY	🟩	3	NORMAL FILE STORAGE & APP SERVERS	SPEED + FAULT TOLERANCE	LAG FOR WRITE OPS, REDUCED STORAGE (BY 1/3)
6	STRIPING & DOUBLE PARITY	🟩	4	LARGE FILE STORAGE & APP SERVERS	EXTRA LEVEL OF REDUNDANCY, HIGH READ PERFORMANCE	LOW WRITE PERFORMANCE, REDUCED STORAGE (BY 2/5)
10 (1+0)	STRIPING & MIRRORING	🟩	4	HIGHLY UTILIZED DATABASE SERVERS	WRITE PERFORMANCE + STRONG FAULT TOLERANCE	REDUCED STORAGE (1/2), LIMITED SCALABILITY

Which RAID is best?

Сравнение уровней RAID

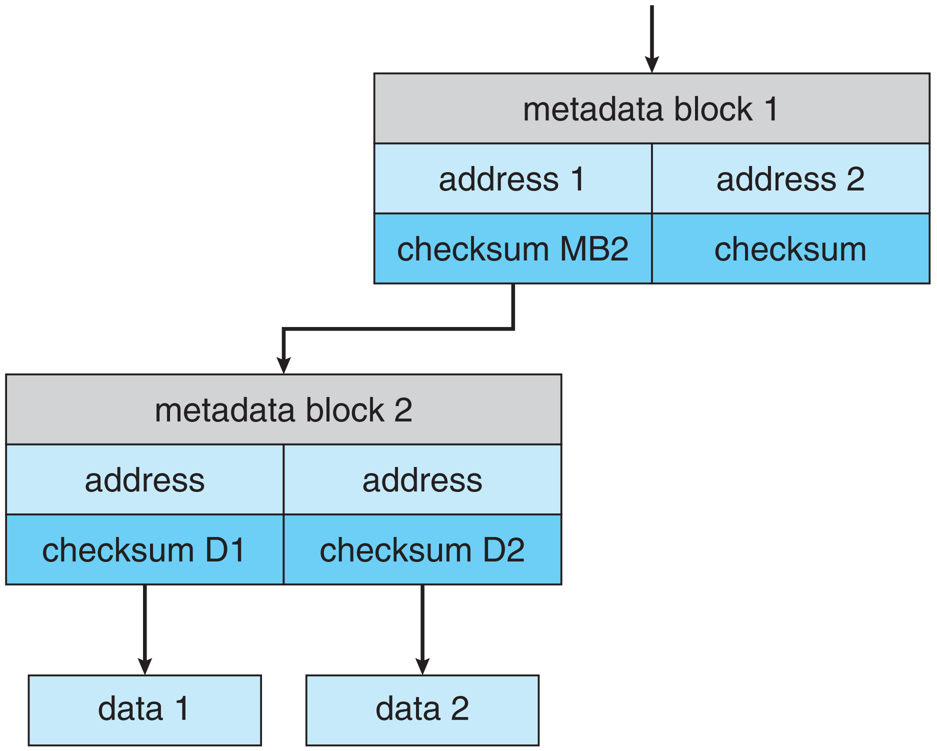


Свойства	RAID 0	RAID 1	RAID 1E	RAID 5	RAID 6	RAID 10	RAID 50
Мин. кол-во дисков	2	2	3	3	4	4	6
Защита данных	Нет	Отказ 1-го диска	Отказ 1-го диска	Отказ 1-го диска	Отказ до 2-ух дисков	Отказ 1-го диска в каждом подмассиве	Отказ 1-го диска в каждом подмассиве
Скорость чтения	Быстро	Быстро	Быстро	Быстро	Быстро	Быстро	Быстро
Скорость записи	Быстро	Средне	Средне	Медленно	Медленно	Средне	Средне
Скорость чтения (degraded)	N/A	Средне	Быстро	Медленно	Медленно	Быстро	Средне
Скорость записи (degraded)	N/A	Быстро	Быстро	Медленно	Медленно	Быстро	Средне
Использование емкости[1]	N*C	N*C/2	N*C/2	(N-1)*C	(N-2)*C	N*C/2	(N-2)*C
Типичное применение	High End рабочие станции, загрузка данных, вывод в реальном времени, временные данные	Операционная система, база данных	Операционная система, база данных	Большие БД, web сервер, архивация	Архив данных, backup, приложения высокой доступности, серверы с высокими требованиями к объему данных	Быстрые БД, серверы приложений	Большие БД, файловые серверы, серверы приложений

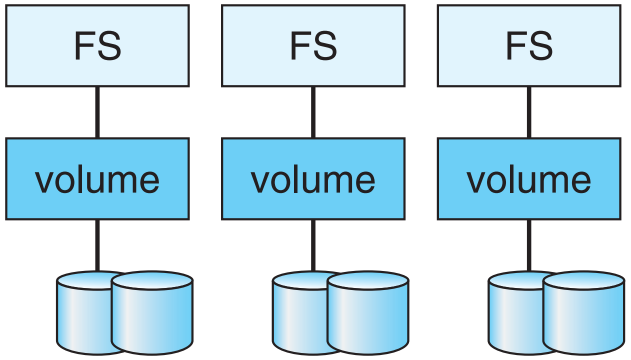
Extensions for Storages

- 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

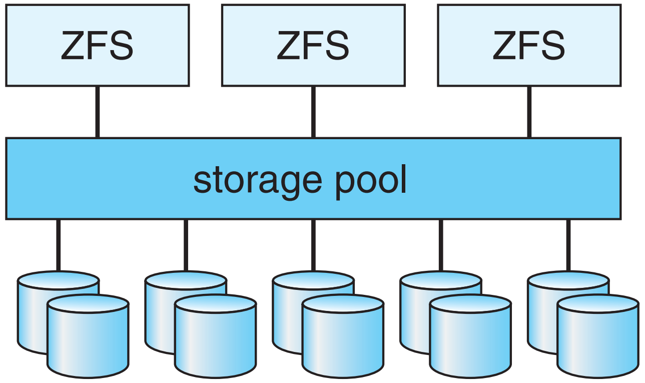
ZFS Checksums All Metadata and Data



Traditional and Pooled Storage



(a) Traditional volumes and file systems.



(b) ZFS and pooled storage.

Stable-Storage Implementation

- Write-ahead log scheme requires stable storage
- Stable storage means data is never lost (due to failure, etc)
- To implement stable storage:
  - Replicate information on more than one nonvolatile storage media with independent failure modes
  - Update information in a controlled manner to ensure that we can recover the stable data after any failure during data transfer or recovery
- Disk write has 1 of 3 outcomes

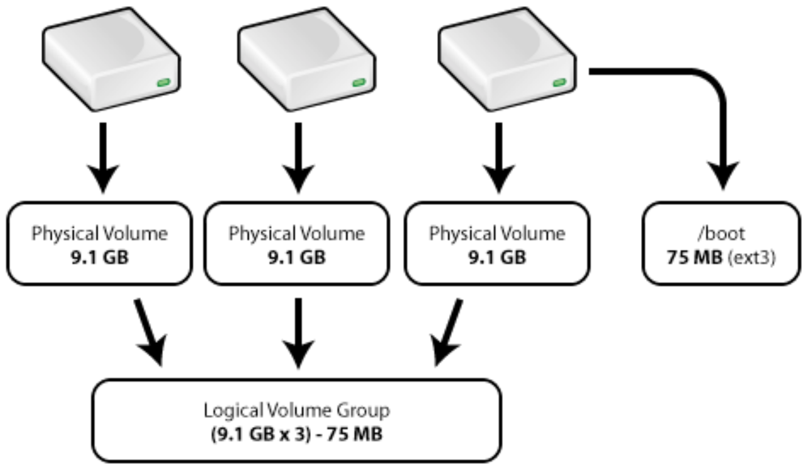
1. **Successful completion -** The data were written correctly on disk
2. **Partial failure -** A failure occurred in the midst of transfer, so only some of the sectors were written with the new data, and the sector being written during the failure may have been corrupted
3. **Total failure -** The failure occurred before the disk write started, so the previous data values on the disk remain intact

- If failure occurs during block write, recovery procedure restores block to consistent state
  - System maintains 2 physical blocks per logical block and does the following:
    1. Write to 1st physical
    2. When successful, write to 2nd physical
    3. Declare complete only after second write completes successfully
  - Systems frequently use NVRAM as one physical to accelerate

Linux Extension: LVM

Logical Volume Group

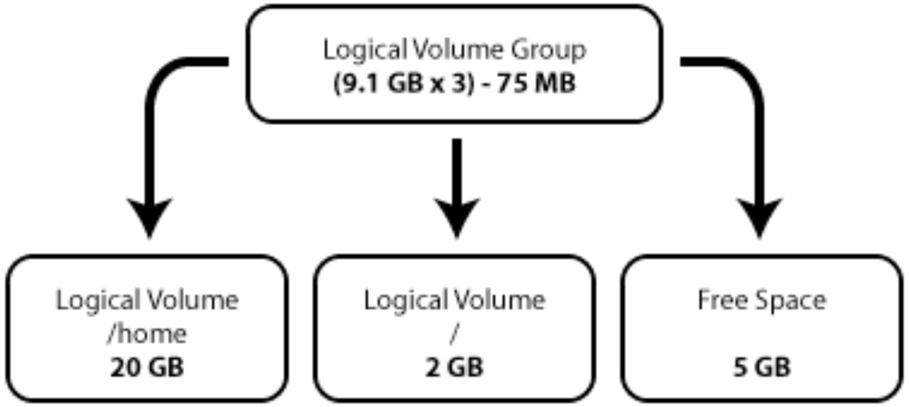
- LVM support must be compiled into the kernel (Example default Red Hat kernel is compiled with LVM support).
- LVM is a method of allocating hard drive space into logical volumes that can be easily resized instead of partitions.
- With LVM, a hard drive or set of hard drives is allocated to one or more *physical volumes*. A physical volume cannot span over more than one drive.
- The physical volumes are combined into *logical volume groups*, with the exception of the /boot/ partition. The /boot/ partition cannot be on a logical volume group because the boot loader cannot read it. If the root (/) partition is on a logical volume, create a separate /boot/ partition which is not a part of a volume group.
- Since a physical volume cannot span over multiple drives, to span over more than one drive, create one or more physical volumes per drive.



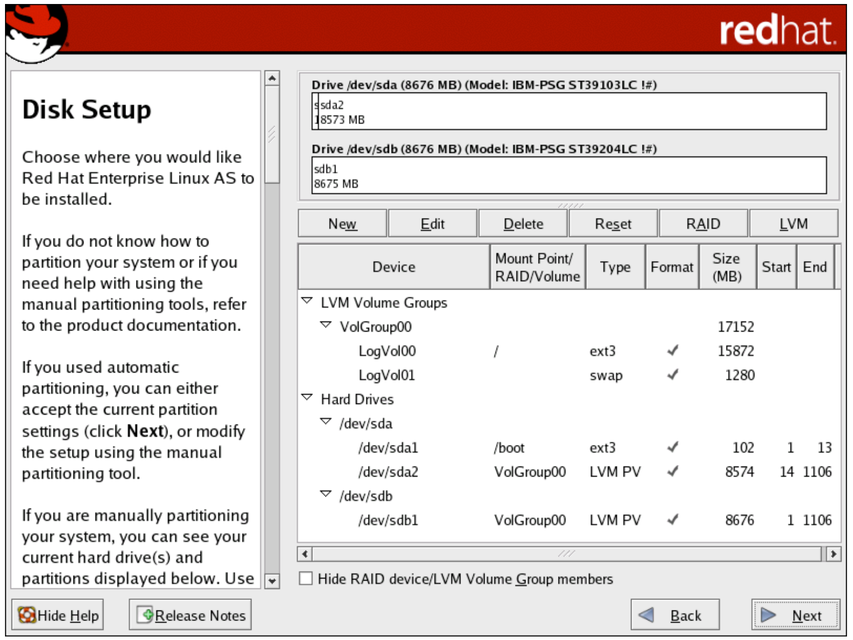
Linux Extension: LVM (Cont.1)

Logical Volumes

- The logical volume group is divided into *logical volumes*, which are assigned mount points, such as /home and / m and file system types, such as ext2 or ext3.
- When "partitions" reach their full capacity, free space from the logical volume group can be added to the logical volume to increase the size of the partition.
- When a new hard drive is added to the system, it can be added to the logical volume group, and partitions that are logical volumes can be expanded.



Automatic LVM Configuration With Two SCSI Drives



Disk Setup

Choose where you would like Red Hat Enterprise Linux AS to be installed.

If you do not know how to partition your system or if you need help with using the manual partitioning tools, refer to the product documentation.

If you used automatic partitioning, you can either accept the current partition settings (click **Next**), or modify the setup using the manual partitioning tool.

If you are manually partitioning your system, you can see your current hard drive(s) and partitions displayed below. Use

Drive /dev/sda (8676 MB) (Model: IBM-PSG ST39103LC !#)
sda2
8573 MB

Drive /dev/sdb (8676 MB) (Model: IBM-PSG ST39204LC !#)
sdb1
8675 MB

Device	Mount Point/ RAID/Volume	Type	Format	Size (MB)	Start	End
LVM Volume Groups						
VolGroup00				17152		
LogVol00	/	ext3	✓	15872		
LogVol01		swap	✓	1280		
Hard Drives						
/dev/sda						
/dev/sda1	/boot	ext3	✓	102	1	13
/dev/sda2	VolGroup00	LVM PV	✓	8574	14	1106
/dev/sdb						
/dev/sdb1	VolGroup00	LVM PV	✓	8676	1	1106

☐ Hide RAID device/LVM Volume Group members

Linux Extension: Disk Quotas

About Disk Quotas

- Disk space can be restricted by implementing disk quotas which alert a system administrator before a user consumes too much disk space or a partition becomes full.
- Disk quotas can be configured for individual users as well as user groups. This possible to make each user a small quota to "personal" files, and more sizable quotas for projects (groups).
- In addition, quotas can be set not just to control the number of disk blocks consumed but to control the number of inodes (data structures that contain information about files in UNIX file systems). Because inodes are used to contain file-related information, this allows control over the number of files that can be created.
- Quotas can be set to control:
  - number of disk blocks
  - number of inodes (numbers of created files)

Configuring Disk Quotas steps:

- Enable quotas per file system by modifying the /etc/fstab file.
- Remount the file system(s).
- Create the quota database files and generate the disk usage table.
- Assign quota policies.

Linux Extension: Disk Quotas (Cont.1)

Enabling Quotas

- As root edit the /etc/fstab file. Add the usrquota and/or grpquota options to the file systems that require quotas.
- In this example, the /home file system has both user and group quotas enabled.
- The following examples assume that a separate /home partition was created during the installation of Linux.

```
/dev/VolGroup00/LogVol00 /         ext3    defaults        1 1
LABEL=/boot              /boot     ext3    defaults        1 2
none                     /dev/pts  devpts  gid=5,mode=620  0 0
none                     /dev/shm  tmpfs   defaults        0 0
none                     /proc     proc    defaults        0 0
none                     /sys      sysfs   defaults        0 0
/dev/VolGroup00/LogVol02 /home     ext3    defaults,usrquota,grpquota  1 2
/dev/VolGroup00/LogVol01 swap      swap    defaults        0 0
.
```

Creating the Quota Database Files

- To create the quota files (aquota.user and aquota.group) on the file system, use quotacheck -c command. For example, if user and group quotas are enabled for the /home file system, create the files in the /home directory: **quotacheck –cug /home**

Assigning Quotas per User

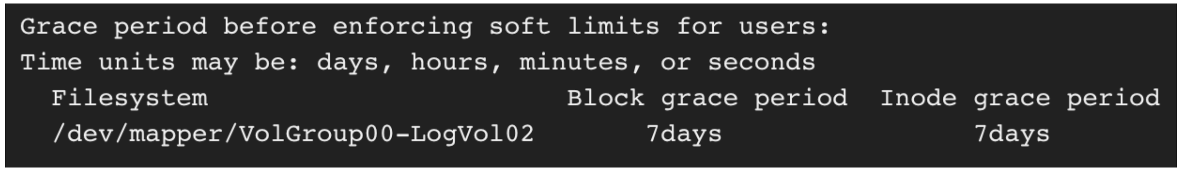
- To configure the quota for a user execute the command as root: **edquota username**
- For example command **edquota testuser** is executed:

```
Disk quotas for user testuser (uid 501):
  Filesystem                blocks       soft       hard     inodes     soft     hard
  /dev/VolGroup00/LogVol02  440436     500000     550000      37418        0        0
```

Linux Extension: Disk Quotas (Cont.2)

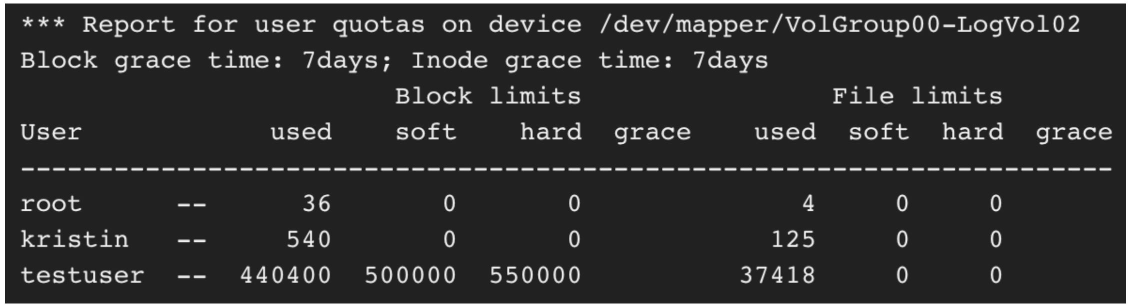
- **Quota format**:
  - The first column is the name of the file system that has a quota enabled for it.
  - The second column shows how many blocks the user is currently using.
  - The next two columns are used to set soft and hard block limits for the user on the file system.
  - The inodes column shows how many inodes the user is currently using.
  - The last two columns are used to set the soft and hard inode limits for the user on the file system.
- **A hard limit** is the absolute maximum amount of disk space that a user or group can use. Once this limit is reached, no further disk space can be used.
- **The soft limit** defines the maximum amount of disk space that can be used. The soft limit can be exceeded for a certain amount of time - *grace period* in seconds, minutes, hours, days, weeks, or months.
- If any of the values are set to 0, that limit is not set. In the text editor, change the desired limits.

For assigning Quotas Grace Period for all File Systems use command edquota –t



```
Grace period before enforcing soft limits for users:
Time units may be: days, hours, minutes, or seconds
  Filesystem                       Block grace period  Inode grace period
  /dev/mapper/VolGroup00-LogVol02       7days                 7days
```

For Reporting of Disk Quotas use repquota utility.
For example, **repquota /home**:



```
*** Report for user quotas on device /dev/mapper/VolGroup00-LogVol02
Block grace time: 7days; Inode grace time: 7days
                        Block limits                File limits
User            used    soft    hard  grace    used  soft  hard  grace
----------------------------------------------------------------------
root      --      36       0       0              4     0     0
kristin   --     540       0       0            125     0     0
testuser  --  440400  500000  550000          37418     0     0
```

END