File Systems - RAID & LVM

Explicación de práctica 4

Sistemas Operativos

Facultad de Informática
Universidad Nacional de La Plata

2019

Agenda

1. File Systems
2. RAID
3. LVM
4. BTRFS

Agenda

- Aplicaciones de computadoras necesitan almacenar y recuperar información
- Tres requerimientos fundamentales para el almacenamiento de la información:
  - Posibilidad de almacenar una gran cantidad de información
  - La información debe permanecer disponible después que finaliza el proceso que la está utilizando
  - Posibilidad de que los procesos accedan concurrentemente a la información
- Información almacenada en archivos (files). En muchas situaciones, la entrada y salida de los procesos es mediante archivos
- File System es la parte del SO que se encarga del manejo de los archivos

Archivo

Un archivo es un conjunto de información o datos relacionados que (probablemente) vivará por mucho tiempo

- Vista del usuario: conjunto de datos persistentes con un nombre
- Vista del SO: colección de bloques de disco
- Tres componentes asociados con un archivo:
  - Nombre
  - Metadata
  - Datos

```
mrobles@mrobles:/$ ls -l sqlDIBigL
-rw------- 1 root root 31903 feb 12  2009 sqlDIBigL
```

File System

Un file system es una abstracción que permite la creación, eliminación, modificación y búsqueda de archivos y su organización en directorios

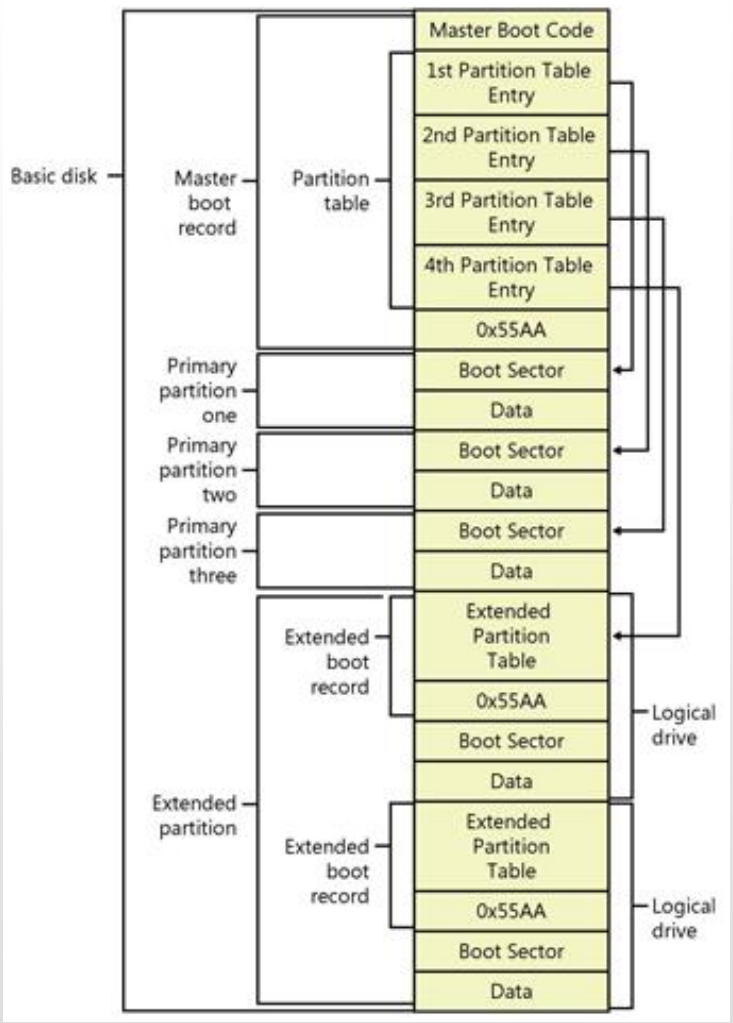
- También administra el control de acceso a los archivos y el espacio en disco asignado a él
- File systems operan sobre bloques de datos (conjunto consecutivo de sectores físicos)
- Archivos son almacenados en una estructura jerárquica de tipo árbol ("/" en Linux, "C" en Windows)
- Define convenciones para el nombrado de los archivos
- File systems usados en discos, CDs, etc.; otros proveen acceso por la red (NFS, SMB, etc.); otros son virtuales (procfs, sysfs)

- Discos pueden ser utilizados completamente por un "file system" o pueden ser "particionado"
- Particiones son subdivisiones de un disco entero. SO las presenta como un dispositivo de bloques (como si fuera el disco entero)
- Están definidas en un área especial del disco llamado "partition table"
- Cada partición contiene un "file system" específico o es una "raw partition"
- Existen varios tipos de tablas de particiones. Dos más utilizadas:
  - **Master Boot Record (MBR)**
  - **Globally Unified Identifier Partition Table (GPT)**

- MBR/BIOS desarrollado para IBM PC usando DOS (1983)
- MBR son los primeros 512 bytes de un dispositivo de almacenamiento (primer sector)
- Contiene el "bootloader" del SO y la tabla de particiones
- BIOS (Basic Input/Output System) es el encargado de llamar al bootloader
- 4 particiones primarias. Particiones extendidas/lógicas.
- Formato CHS (Cylinder, Head, Sector): inicio/fin de cada partición. C:0..1023, H:0..255, S: 1..63. Con sectores de 512B, tamaño máximo del disco duro es de 7,844GB
- "Logical Block Addressing (LBA)" permitió acceder a discos de mayor tamaño
- LBA es un esquema de direccionamiento lineal
- ATA-6, año 2006, introdujo LBA-48 bits (CHS obsoleto)

- MBR divide los primeros 512 bytes:
  - "Master Boot Code": también llamado *bootloader*, ocupa los primeros 440 bytes
  - "Disk signature": 6 bytes. Identifica el disco al SO
  - "Primary" o "Master Partition Table": 64 bytes. 16 bytes por cada partición
  - "Signature Word": 0xAA55
- Bootloader encuentra y pasa el control al "Boot Sector" de la partición activa
- LILO, GRUB, SysLinux son bootloader comunes en Linux
- Particiones lógicas no se encuentran en la MPT sino en una "Extended Partition Table (EPT)" separada.
- EPT misma forma que la MPT, pero usa dos entradas
- Si existen múltiples particiones lógicas se crea una lista linkeada de múltiples EPTs ("daisy chained")

File Systems - MBR

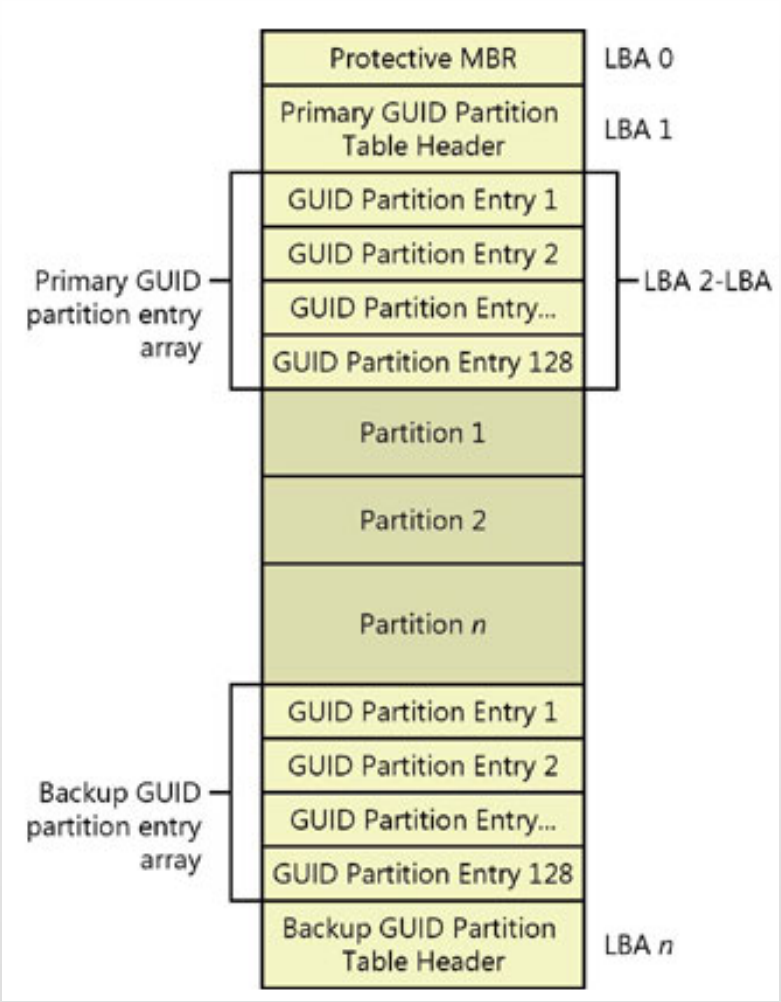


https://technet.microsoft.com

- Introducida por Intel a finales de los 90 para los Intel Itanium como parte del EFI: “Extensible Firmware Interface”.
- Cedida por Intel al UEFI Forum: “Unified Extensible Firmware Interface”
- No más particiones extendidas/lógicas. 128 particiones por defecto.
- LBA de 64 bits $=>$ 2^{64} sectores de 512 bits. Soporta hasta 8ZiB (ó 9,4ZB)
- Backup de la GPT final del disco
- CRC para proteger el contenido de toda la GPT
- Utiliza GUID (“Globally Unique IDentifier”) para referenciar los discos/particiones

- **Protective MBR:** primer sector del disco (LBA 0, *SystemID* = *0xEE*)
- **GUID Partition Table Header:**
  - Tamaño mínimo de 16.384 bytes (permiten 128 particiones)
  - Checksum CRC32 que cubre este sector y toda la tabla de particiones
  - Ubicación (nro. LBA) de las 2 GPT (original y backup). También indica su tamaño
  - GUID del disco (en UNIXes se lo conoce como UUID)
- Cada entrada en la tabla de particiones es de 128 bytes
- Particiones son identificadas por su propio GUID

File Systems - GPT

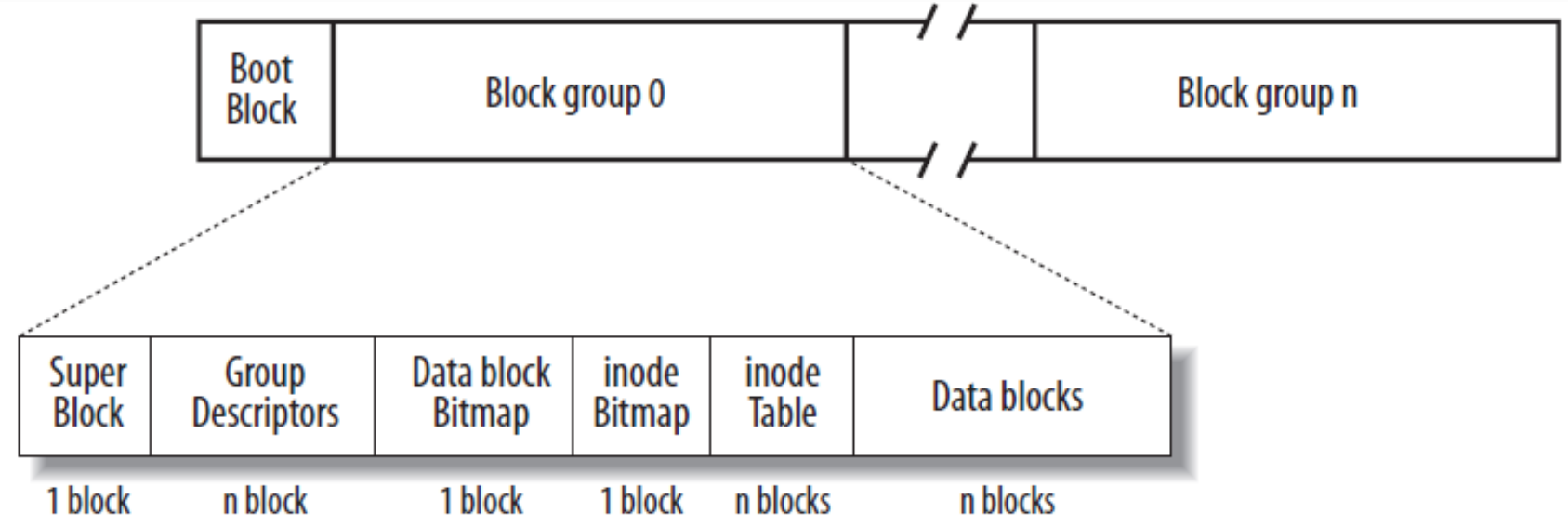


https://technet.microsoft.com

- Principales herramientas en Linux para particionar:
  - parted
  - gparted
  - fdisk
  - gdisk
- Se usará parted porque soporta tanto MBR como GPT

- Second Extended Filesystem (Ext2). Introducido en 1994
- Primer sector de la partición no es administrado por Ext2
- Dividido en "block groups" de igual tamaño (excepto el último)
- "Block groups" reducen la fragmentación y aumentan la velocidad de acceso
- "Superblock" y "Group Descriptors Table" replicados en block groups para backup
- Por cada bloque existe un "Group Descriptor"
- "Block Bitmap" e "Inode Bitmap" indican si un bloque de datos o inodo está libre u ocupado
- Tabla de inodos consiste de una serie de bloques consecutivos donde cada uno tiene un número predefinido de inodos. Todos los inodos de igual tamaño: 128 bytes (por default)

- Esquema de un file system Ext2

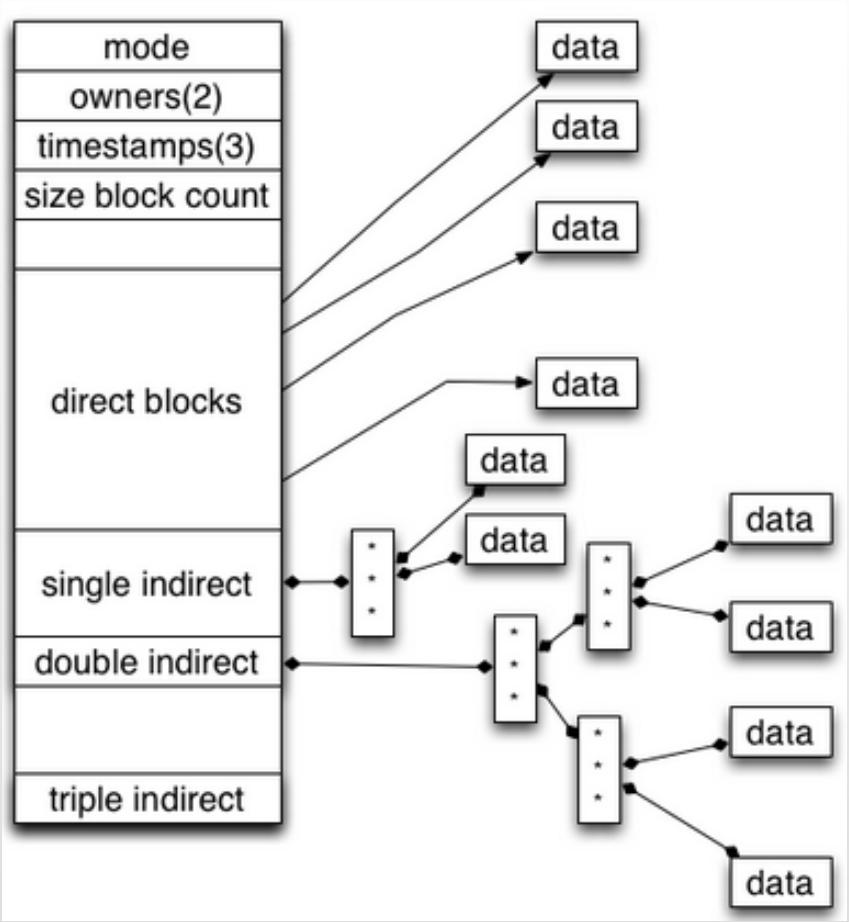


- "Superblock" y "Group Descriptors" no es necesario replicarlo en todos los grupos
- Solo "superblock" del grupo 0 se lee y se modifica

Fuente: OReilly - Understanding the Linux Kernel - Daniel Bovet y Marcos Cesati

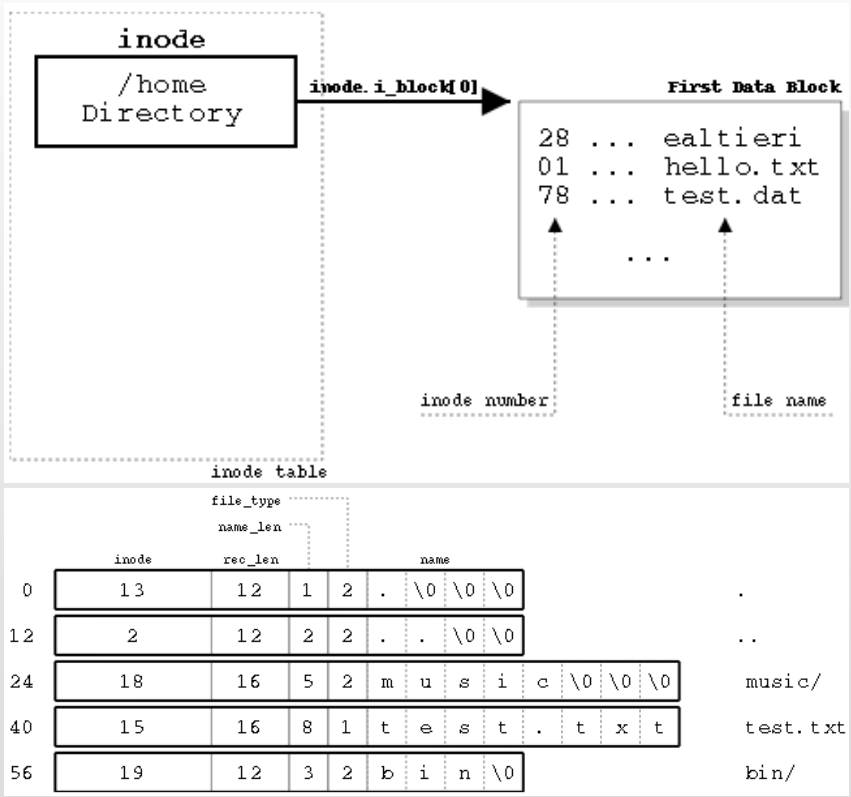
- Cada archivo en el file system es representado por un inodo (index node)
- Inodos contienen metadata de los archivos y punteros a los bloques de datos
- Metadata: permisos, owner, grupo, flags, tamaño, número bloques usados, tiempo acceso, cambio y modificación, etc.
- Nombre del archivo no se almacena en el inodo
- Atributos extendidos, como las ACLs, se almacenan en un bloque de datos (campo en el inodo llamado "i_file_acl))
- Datos se almacenan en bloques de 1024, 2048 ó 4096 bytes. Elegible al momento de generar el file system. No se puede modificar

- Estructura de un inodo:



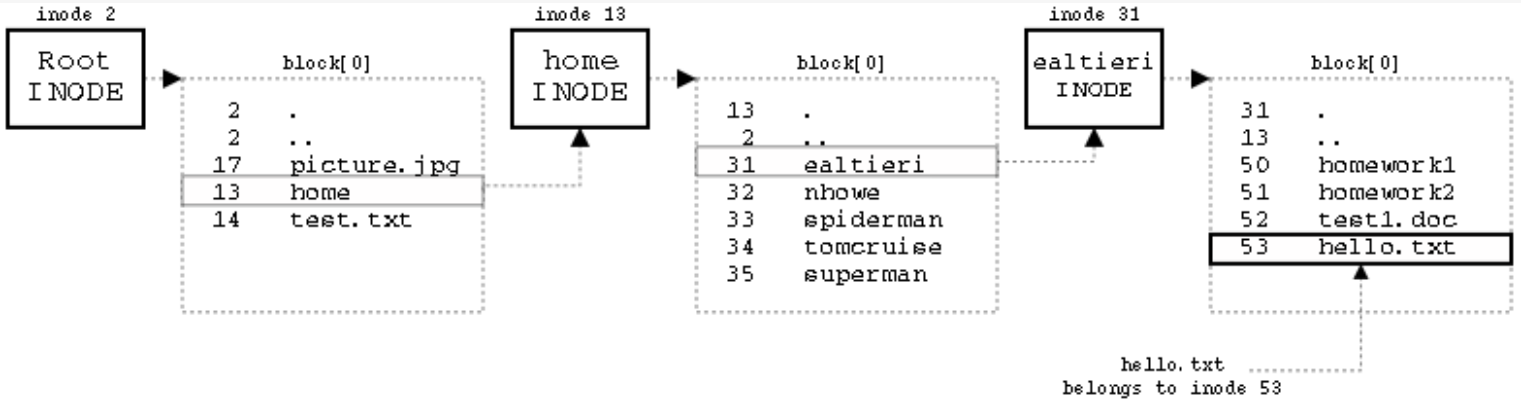
Fuente: OReilly - Understanding the Linux Kernel - Daniel Bovet y Marcos Cesati

- Ext2 implementa directorios como un tipo especial de archivo cuyo bloques de datos almacenan el nombre del archivo y su inodo correspondiente



Fuente: http://cs.smith.edu/ nhowe/262/oldlabs/ext2.html

- ¿Cómo se encuentra un archivo en el file system?
- int fd = open(/home/ealtieri/hello.txt", O_RDONLY);



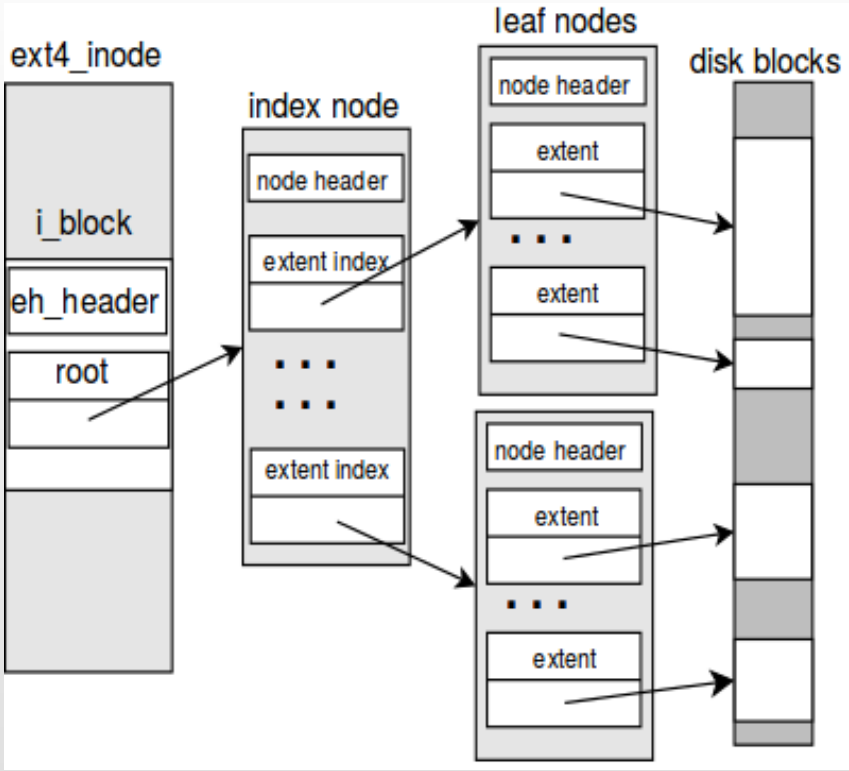
- Directorio raíz siempre se ubica en el inodo 2

Fuente: http://cs.smith.edu/ nhowe/262/oldlabs/ext2.html

- Ext3, Third Extended FileSystem, es la evolucion de Ext2
- Introducido en 2001. Disponible desde la version de kernel 2.4.15
- Su principal mejora se basa en la incorporación del "journaling", que permite reparar posibles inconsistencias en el file system
- Journaling: journal, ordered, writeback
- Es compatible con ext2
- Tamaño máximo de archivo 2TB. Tamaño máximo de File System: 32 TB (ambos igual que en ext2)
- Cantidad máxima de subdirectorios: 32000

- Ext4, Fourth Extended FileSystem, es la evolucion de Ext2
- Introducido en 2006. Disponible desde la version de kernel 2.6.19
- Sistema de archivos de 64 bits (FS de 1EB, files de 16TB)
- Cantidad máxima de subdirectorios: 64000 (extendible)
- Tamaño inodo: 256 bytes (timestamps más precisos, ACLs).
- Uso de extents: descriptor que representa un rango contiguo de bloques físicos
- Cada extent puede respresentar 2^{15} bloques (128MB con bloques de 4KB, 4 extents por inodo)
- Para archivos más grandes se utiliza un árbol de extents
- Mejor alocación de bloques para disminuir la fragmentación e incrementar el throughput: “persistent preallocation”, “delay and multiple block allocation”

- Estructura de un inodo con extent:



- Desarrollado por SGI (Silicon Graphics Inc.) para IRIX (su versión de UNIX)
- En 2000, SGI lo liberó bajo una licencia de código abierto
- Incorporado a Linux desde la versión 2.4.25
- Red Hat 7.0 lo incluye como su FS default (CentOS desde la versión 7.2)
- File System de 64 bits (16 EB max. file system, 8EB max. file size)
- FS dividio en regiones llamadas “allocation groups”. Uso de extents. Inodo asignados dinámicamente.
- Journaling (primer FS de la familia UNIX en tenerlo)
- Mayor espacio para atributos extendidos (hasta 64KB)
- Contra: no es posible achicar un FS de este tipo

- ProcFS es un pseudo-filesystem montado comúnmente en el directorio /proc
- Provee una interface a las estructuras de datos del kernel
- Presenta información sobre procesos y otra información del sistema en una estructura jerárquica de “files”
- No existe en disco, el kernel lo create en memoria (generalmente tamaño 0 de los “files”)
- Mayoría de los “files” de solo lectura, aunque algunos pueden ser modificados (/proc/sys)
- echo 1 > /proc/sys/net/ipv4/ip_forward (o con el comando sysctl)
- /proc/pid: dir. con información del proceso “pid”
- /proc/filesystems: lista los FS soportados por el kernel
- /proc/meminfo: información del uso de memoria física y swap

```
mrobles@mrobles:~$ ls /proc
1      12858  175   1890   20368  2085   211    22    2387  2622  39   55   67   82   94          fb           modules      timer_stats
10     12877  1750  1892   20375  20852  2111   2201  2395  27    394  56   68   83   947         filesystems  mounts       tty
101    12892  1758  18944  20376  20861  21115  2207  24    270   40   57   6836 8524 95          fs           mtrr         uptime
1021   13     176   19     20444  20889  212    2219  2432  2768  405  59   6844 8793 980         interrupts   net          version
10242  14     177   192    20483  2091   21233  2221  2442  28    41   598  6875 9    982         iomem        pagetypeinfo version_signature
```

```
mrobles@mrobles:~$ ls -l /proc
total 0
dr-xr-xr-x  9 root    root    0 may 19 11:41 1
dr-xr-xr-x  9 root    root    0 may 19 11:44 10
dr-xr-xr-x  9 root    root    0 may 19 11:44 101
dr-xr-xr-x  9 root    root    0 may 19 11:44 1021
dr-xr-xr-x  9 mrobles mrobles 0 may 20 10:47 10242
dr-xr-xr-x  9 root    root    0 may 19 11:44 1031
dr-xr-xr-x  9 root    root    0 may 19 11:44 1038
dr-xr-xr-x  9 root    root    0 may 19 11:44 11
dr-xr-xr-x  9 root    root    0 may 20 18:19 11320
```

- Con el paso del tiempo, /proc se convirtió en un verdadero desorden
- En Linux 2.6 se implementó un nuevo sistema de archivos virtual llamado “Sysfs”
- SysFS exporta información sobre los dispositivos de hardware y sus controladors desde el kernel hacia el espacio del usuario
- También permite la configuración de parámetros
- SysFS se monta en /sys

```
mrobles@mrobles:~$ ls -l /sys
total 0
drwxr-xr-x   2 root root 0 may 22 13:31 block
drwxr-xr-x  38 root root 0 may 22 13:31 bus
drwxr-xr-x  63 root root 0 may 22 13:31 class
drwxr-xr-x   4 root root 0 may 22 13:31 dev
drwxr-xr-x  14 root root 0 may 22 13:31 devices
drwxr-xr-x   5 root root 0 may 22 13:31 firmware
drwxr-xr-x   7 root root 0 may 22 13:31 fs
drwxr-xr-x   2 root root 0 may 22 13:31 hypervisor
drwxr-xr-x  10 root root 0 may 22 13:31 kernel
drwxr-xr-x 199 root root 0 may 22 13:31 module
drwxr-xr-x   2 root root 0 may 22 13:31 power
mrobles@mrobles:~$ ls -l /sys/block/
total 0
lrwxrwxrwx 1 root root 0 may 19 11:41 dm-0 -> ../devices/virtual/block/dm-0
lrwxrwxrwx 1 root root 0 may 19 11:44 dm-1 -> ../devices/virtual/block/dm-1
lrwxrwxrwx 1 root root 0 may 22 13:31 loop0 -> ../devices/virtual/block/loop0
lrwxrwxrwx 1 root root 0 may 22 13:31 loop1 -> ../devices/virtual/block/loop1
lrwxrwxrwx 1 root root 0 may 22 13:31 loop2 -> ../devices/virtual/block/loop2
lrwxrwxrwx 1 root root 0 may 22 13:31 loop3 -> ../devices/virtual/block/loop3
```

```
mrobles@mrobles:~/git-repos/contenidos/explicaciones/so/practica5$ cat /sys/class/block/sda/queue/logical_block_size
512
mrobles@mrobles:~/git-repos/contenidos/explicaciones/so/practica5$ cat /sys/class/block/sda/queue/physical_block_size
4096
```

Agenda

- Problemas:
  - ¿Qué pasa si se rompe un disco? ¿Hay backup?
  - Si se llena un disco, ¿compro un disco más grande?
  - Velocidad de acceso. Operaciones I/O muy lentas
- Solución:
  - RAID - Redundant Array of Independent Disks (originalmente llamado Redundant Array of Inexpensive Disks)

RAID - Introducción (cont.)

- RAID es una técnica que permite usar múltiples discos en forma conjunta con el fin de construir un sistema de discos más rápido, más grande y más confiable
- Idea propuesta en 1988 en la Universidad de Berkley por los profesores David Patterson y Randy Katz y el alumno Garth Gibson: "A case for Redundant Arrays of Inexpensive Disks"
- Originalmente definía los niveles: 1,2,3,4,5
- En 1989, "Disk System Architectures for High Performance Computing" introdujo un nuevo nivel: RAID 6
- Ventajas sobre un solo disco (depende del nivel de RAID):
  - Incremento de la perfomance
  - Mayor capacidad
  - Aumento de la confiabilidad

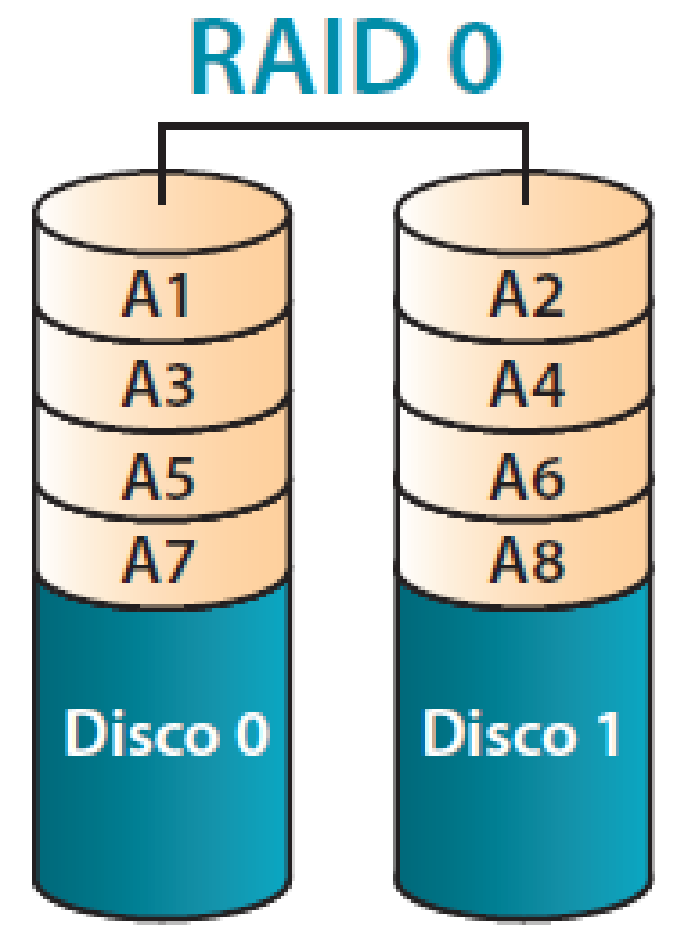
RAID - Introducción (Cont.)

- RAID provee las anteriores ventajas en forma transparente
- Para los file systems es un arreglo lineal de bloques que pueden ser leídos y escritos
- Internamente debe calcular que disco/s debe acceder para completar la solicitud
- Cantidad de accesos físicos de I/O depende del nivel de RAID que se está utilizando
- Diseñados para detectar y recuperarse de determinados fallos de discos
- **Spare Disks:** discos disponibles para reemplazo de discos en falla
- Solo se usan los niveles 0,1,5 y 6 (y alguna combinación entre ellos)

RAID Level 0 - Striping

- No es un nivel de RAID en absoluto. No existe redundancia.
- Array de discos con "striping" a nivel de bloque
- Necesita 2 ó más discos para conformarse
- Capacidad del RAID: sumatoria de la capacidad de los discos participantes
- Sirve como "upper-bound" en cuanto a performance y capacidad
- Si falla un disco, los datos de todos los discos se vuelven inaccesibles
- **Disk Striping:** proceso de dividir datos en bloques y esparcirlos en múltiples dispositivos de almacenamiento

RAID Level 0 (cont.)



Fuente: http://www.dlink.com

- Sólo usado cuando se hace **striping**
- Datos se dividen en pedazos y se escriben en distintos discos

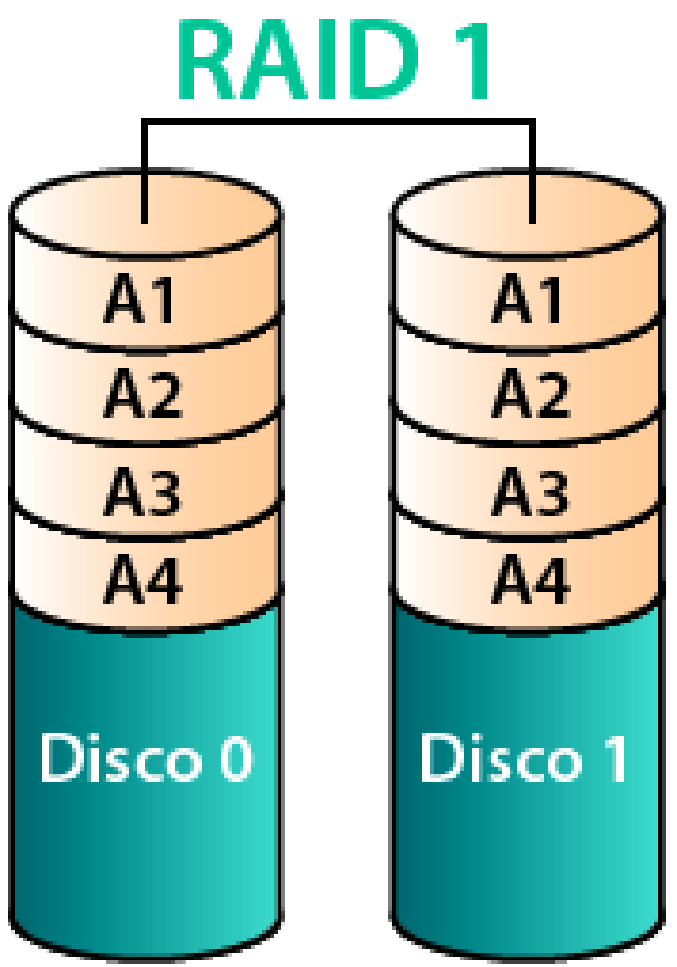
Disk 0	Disk 1	Disk 2	Disk 3
0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Disk 0	Disk 1	Disk 2	Disk 3	
0	2	4	6	chunk size:
1	3	5	7	2 blocks
8	10	12	14	
9	11	13	15	

RAID Level 1 - Mirroring

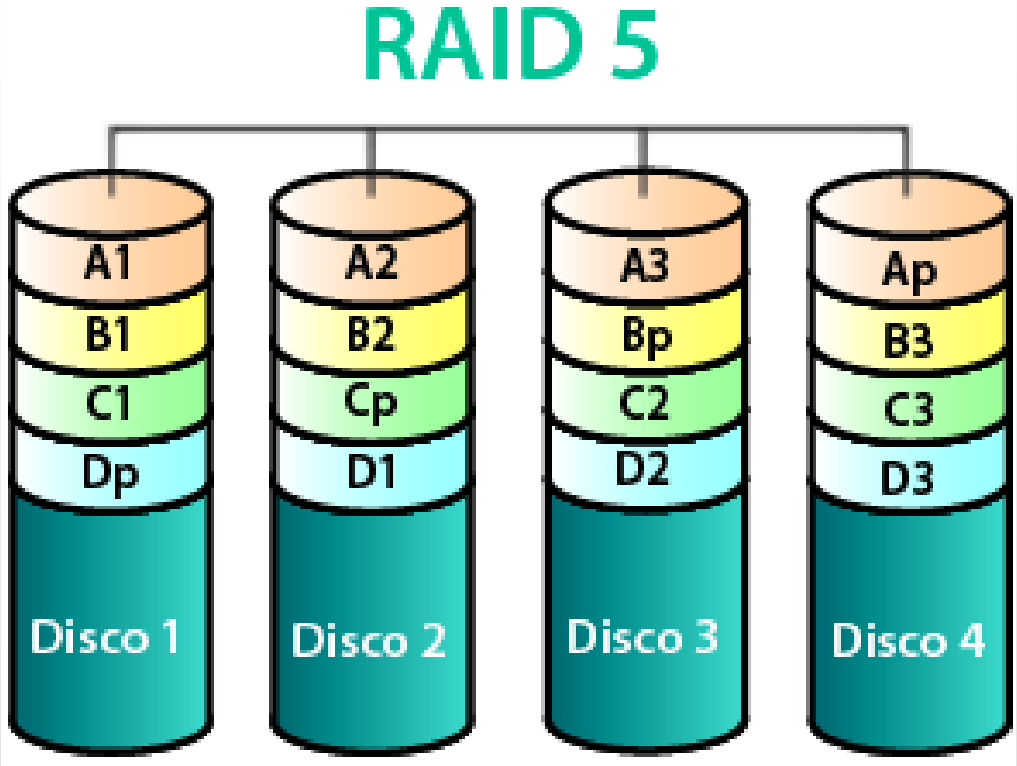
- Asegura redundancia mediante el mirroring (espejado) de datos
- No hay striping de datos
- Almacena datos duplicados en discos separados o independientes
- Mínimo de 2 discos. Trabaja con pares de discos
- Ineficiente por la escritura en espejo
- Desperdicio del 50 % de la capacidad total

RAID Level 1 (cont.)



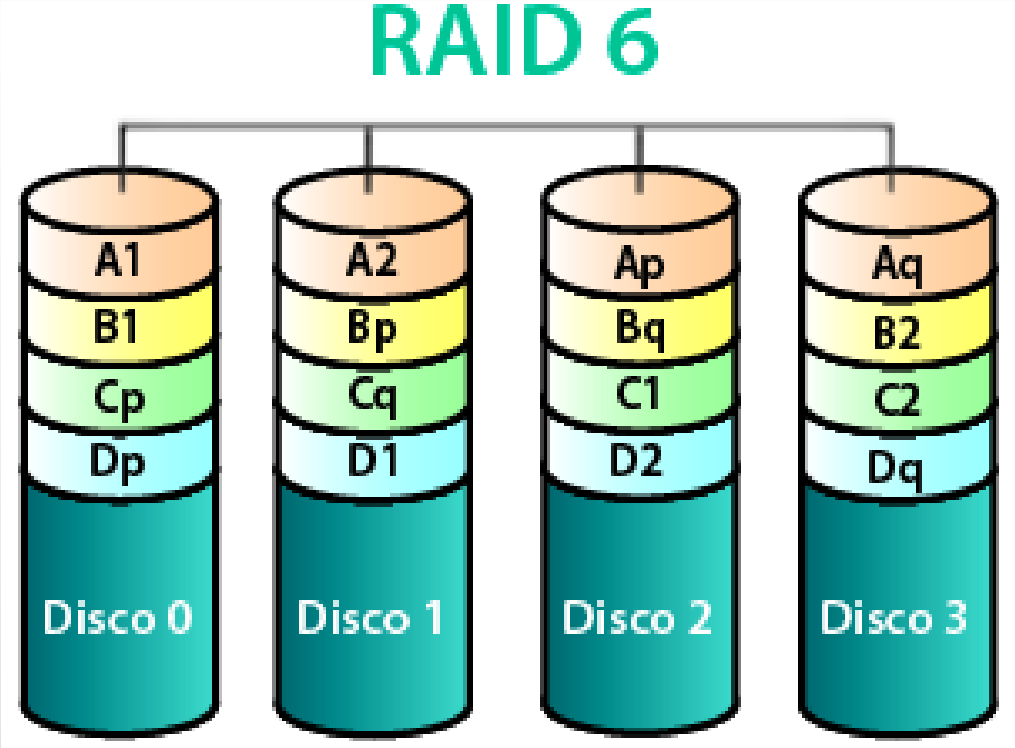
Fuente: http://www.dlink.com

- Striping a nivel de bloque y paridad distribuida
- Una de las implementaciones más utilizadas
- Distribuye la información de paridad entre todos los discos del array
- 3 discos requeridos como mínimo
- Alto rendimiento. No hay cuello de botella
- No ofrece solución al fallo simultáneo de discos



Fuente: http://www.dlink.com

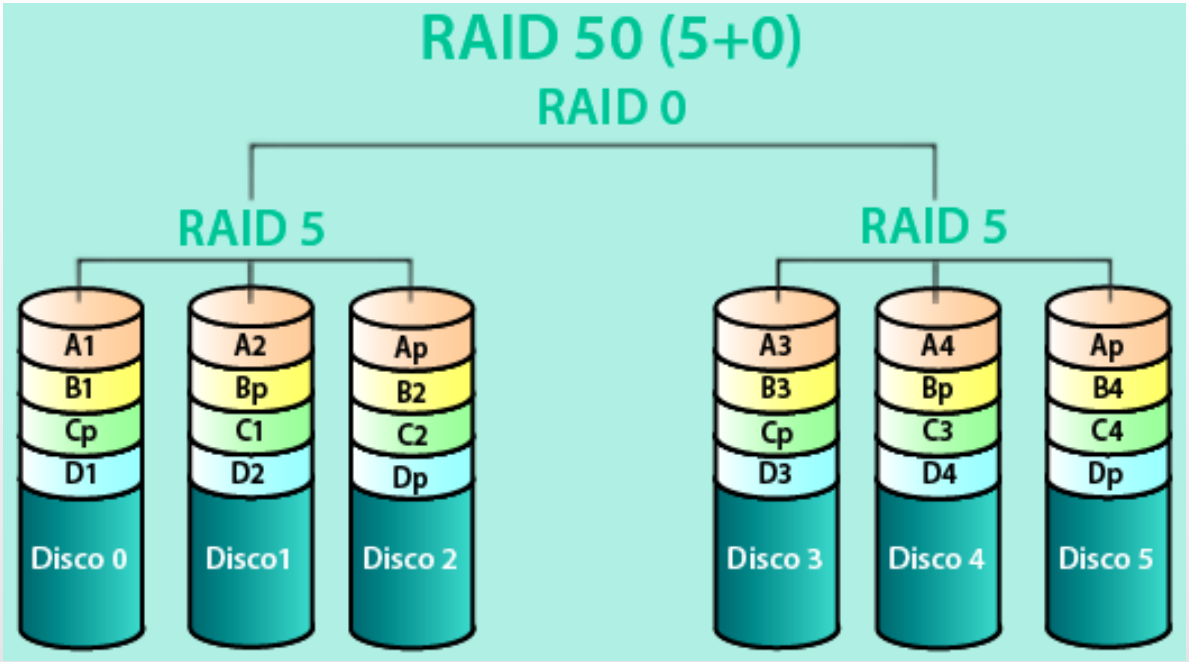
- Striping a nivel de bloque y doble paridad distribuida
- Recomendado cuando se tienen varios discos
- Requiere 4 discos como mínimo
- Alta tolerancia a fallos (hasta dos discos)
- Operaciones de escrituras más lentas debido al calculo de la doble paridad



Fuente: http://www.dlink.com

- Es posible combinar los distintos niveles de RAID
  - RAID 0+1: Mirror of Striped Disks
  - RAID 10(1+0): Stripe of Mirrored Disks
  - RAID 50(5+0)
  - RAID 100
  - etc.

RAID - Niveles híbridos - RAID 50



Fuente: http://www.dlink.com

Configurar RAID Level 5

- Particionar un disco. ¿Cuántas particiones necesarias cómo mínimo?
- Utilizar la herramienta parted para particionar el disco

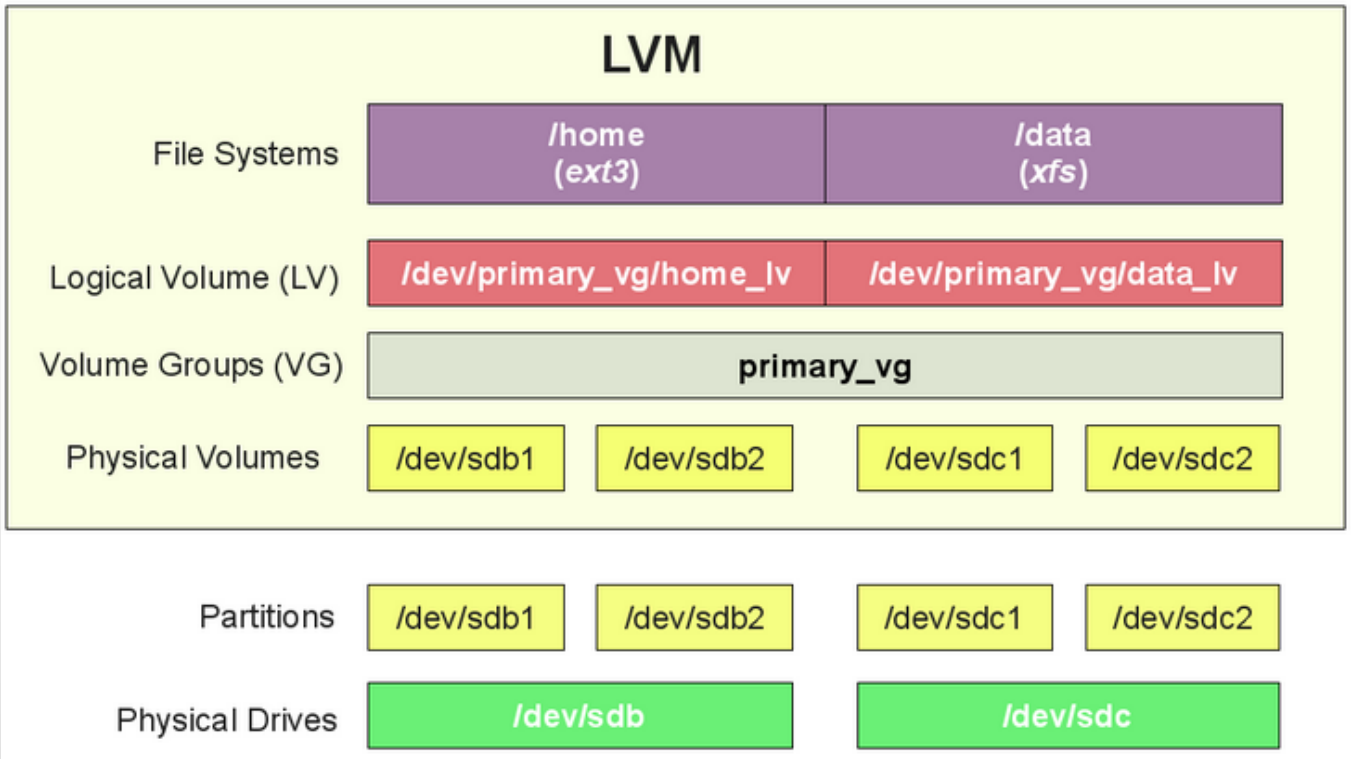
```
mrobles@2sy169:~/Desktop$ sudo parted /dev/sda
GNU Parted 2.3
Usando /dev/sda
¡Bienvenido/a a GNU Parted! Teclee «help» para ver la lista de órdenes.
(parted) print
Modelo: ATA SAMSUNG HD161HJ (scsi)
Disco /dev/sda: 160GB
Tamaño de sector (lógico/físico): 512B/512B
Tabla de particiones. msdos

Numero  Inicio  Fin     Tamaño  Tipo      Sistema de archivos  Banderas
 1      32,3kB  31,7GB  31,7GB  primary   ntfs                 arranque
 2      31,7GB  94,6GB  62,9GB  primary   ntfs
 3      94,6GB  133GB   38,1GB  extended
 5      94,6GB  95,6GB  1003MB  logical   ext3
 6      95,6GB  131GB   35,0GB  logical   ext3
 7      131GB   133GB   2048MB  logical   linux-swap(v1)
 4      133GB   160GB   27,4GB  primary   ext3

(parted)
```

Agenda

- ¿Qué tamaño le voy a dar a una partición?
- Logical Volume Management (LVM) provee un método más flexible que los convencionales esquemas de particionamiento para alocar espacio en los dispositivos de almacenamiento masivo
- Escrito en 1998 por Heinz Mauelshagen (basó el diseño en el existente en HP-UX)
- Actualmente está en la versión 2
- Permite una administración "on-line" de los discos de almacenamiento agregando una capa adicional al subsistema de entrada del kernel
- Snapshots: backup on-line de las particiones

- Principales componentes de LVM:
  - **Physical Volume (PV):** dispositivos físicos o particiones que serán utilizados por LVM
  - **Volume Group (VG):** grupo de PVs. Representa el "data storage"
  - **Logical Volume (PV):** cada una de las partes en las que se dividen los VGs
  - **Physical Extent (PE):** unidades direccionables en las que LVM divide cada PV
  - **Logical Extent (LE):** unidad de alocación básica en los LVs

LVM - Componentes (cont.)

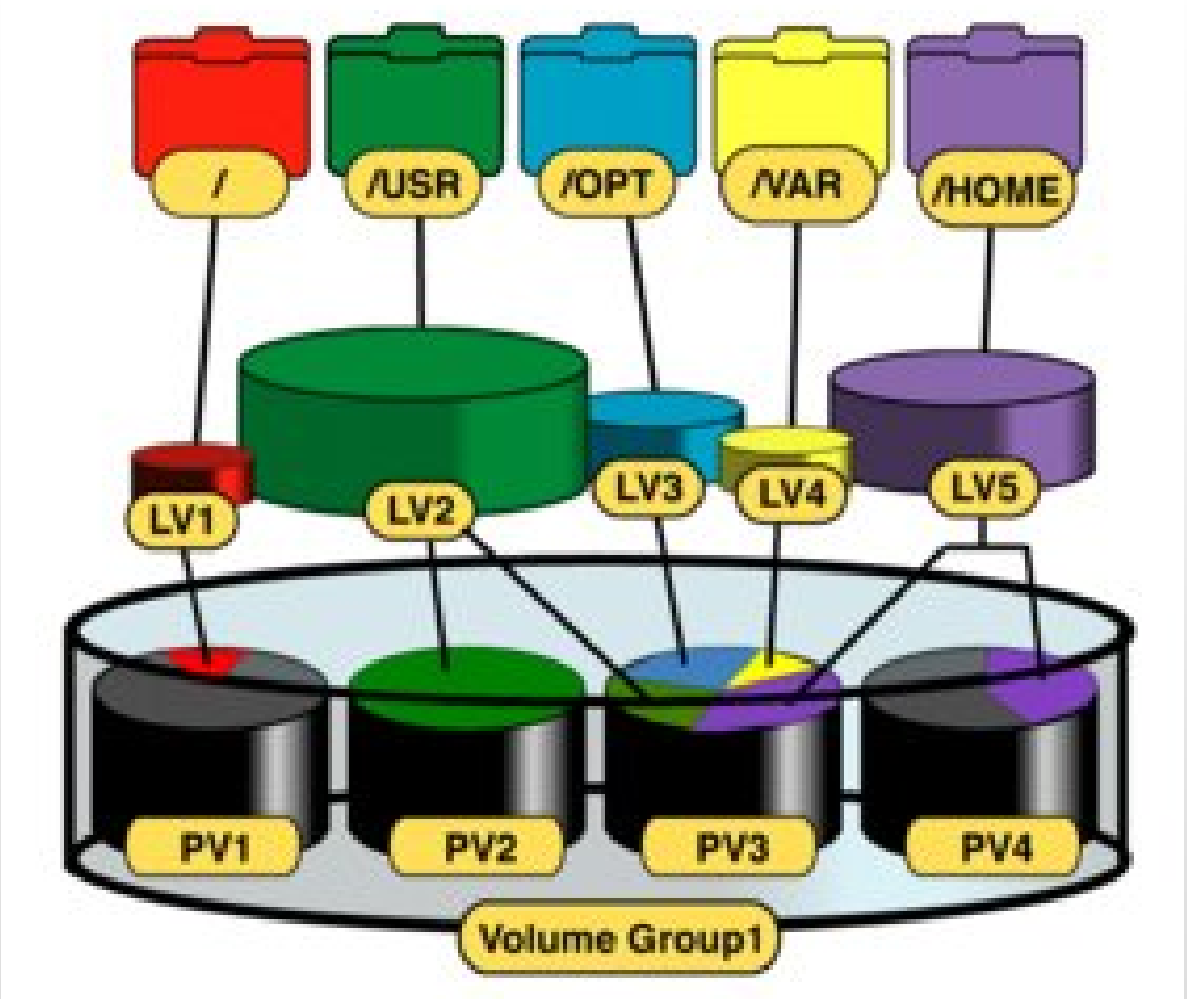


- Generar 3 particiones (sean /dev/sda5, /dev/sda6 y /dev/sda7)
- Configurar estas particiones para que pueden trabajar por LVM
- *pvcreate /dev/sda5 /dev/sda6 /dev/sda7*
- *pvscan* y *pvdisplay* para ver el estado de los PVs

- Siguiente paso: generar un VG en el PV creado en el paso anterior
- *vgcreate so_test /dev/sda5 /dev/sda6 /dev/sda7*
- *vgscan* y *vgdisplay* para ver el estado de los PVs
- Es posible generar varios VG dentro de un PV

- Generar particiones lógicas en el VG generado en el paso anterior
- Hay que indicarle el tamaño. Dos formas:
  - *lvcreate -L 120G -n lv_test so_test*
  - *lvcreate -l 25 -n lv_test so_test*
- *lvscan* y *lvdisplay* para ver el estado de los PVs
- Formatear el LV: *mkfs.ext3 /dev/so_test/lv_test*
- Montar el LV en un punto de montaje

LVM - Componentes (cont.)



LVM - Snapshots

- Problemas al actualizar una aplicación o el SO, o al realizar backups
- No todos los filesystems tienen la capacidad de realizar un snapshot "online"
- *LVM Snaphost:* es una "Point-In-Time" copia de un volumen lógico
- Contiene metadata y los bloques de datos de un LV origen que hayan sido modificados desde que se generó el snapshot
- Se crean instantáneamente y persisten hasta que se los elimina
- "Copy-on-Write (CoW)": datos son copiados al snapshot cuando se modifican
- Calcular cuidadosamente el tamaño del snapshot
- Snapshot es un Logical Volume dentro del Volume Group

Agenda

- B-Tree File System diseñado en Oracle. Licencia GPL
- Comenzó en 2007, estable en Agosto de 2014
- Introducido en Linux kernel 2.6.29, Marzo de 2009
- Participan, o han participado: Red Hat, Fujitsu, Intel, SUSE, etc.
- En 2105, SUSE Linux Enterprise Server 12 lo adoptó como su file system default
- En Agosto de 2017, las *releases notes* de Red Hat Enterprise Linux 7.4 indicaron que dejaba de soportarlo. Nuevo proyecto: **Stratis Proyect**
- Sistema de archivos basado en Copy-On-Write (COW)

- Tamaño máximo de volumen: 16EiB. Tamaño máximo de archivo: 16EiB
- Cantidad máxima de archivos: 2^{64}
- Utilización de extents
- Agregar o remover dispositivo de bloques, agrandar o achicar volúmenes, defragmentación online
- Alocación dinámica de inodos
- Soporte integrado de múltiples dispositivos (RAID0, RAID1, RAID5, RAID6...)
- Checksum de datos y metadatos (CRC32). Compresión (zlib, LZO...). Scrub
- Subvolúmenes. Snapshots.

Copy on Write (CoW) en BTRFS

- Por default, usada en todas las escrituras al file system (puede ser deshabilitada)
- Nuevos datos: creados como siempre
- Modificación datos: copiados en un nuevo espacio (no se sobreescriben), luego se modifican los metadatos. No es necesario el *journaling*
- cp –relinks o btrfs subvolume snapshot no duplican datos
- Es posible la deduplicación, pero no se realiza on-line. Herramientas adicionales.
- Técnica usada por bases de datos, en memoria virtual, etc.

Subvolúmenes

- No es necesario crear particiones: crear un *Storage Pool en el cual se crean subvolúmenes*
- Un subvolumen puede ser accedido como un directorio más o puede ser montado como si fuese un dispositivo de bloques (pero no lo es!!)
- Cada subvolumen puede ser montando en forma independiente (y pueden ser anidados)
- No pueden ser formateados con un filesystem diferente
- Existe un subvolumen en cada filesystem BTRFS llamado *top-level subvolume*
- Es posible limitar la capacidad de cada subvolumen (qgroup/quota)
- No pueden extenderse a otro BTRFS filesystem

Algunos comandos

- *mkfs.btrfs /dev/sdb:* crea un file system BTRFS en una partición
- *mkfs.btrfs /dev/sdb /dev/sdc:* crea un file system BTRFS sobre dos particiones
- *mkfs.btrfs -d raid1 -m raid1 /dev/sdb /dev/sdc:* crea un RAID1 espejando esas dos particiones
- *btrfs device add /dev/sdd /mnt/disco1:* agrega un nuevo dispositivo de bloques a un file system montado
- *btrfs filesystem show o btrfs filesystem df:* para ver las particiones de tipo BTRFS
- *btrfs-convert /dev/sdXX:* convierte un file system Ext3/4 a BTRFS