马哥教育
IT人的高薪职业学院

磁盘存储和文件系统

讲师：王晓春

本章内容

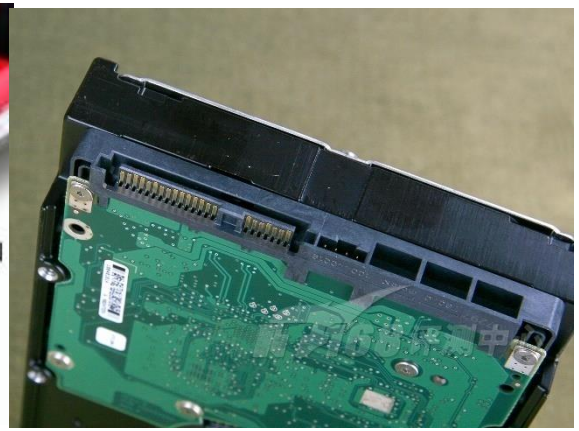
- 磁盘结构
- 分区类型
- 管理分区
- 管理文件系统
- 挂载设备
- 管理虚拟内存
- RAID管理
- LVM管理
- LVM快照

设备文件

- I/O Ports: I/O设备地址
- 一切皆文件：open(), read(), write(), close()
- 设备类型：
  - 块设备：block，存取单位“块”，磁盘
  - 字符设备：char，存取单位“字符”，键盘
- 设备文件：关联至一个设备驱动程序，进而能够跟与之对应硬件设备进行通信
- 设备号码：
  - 主设备号：major number, 标识设备类型
  - 次设备号：minor number, 标识同一类型下的不同设备

硬盘接口类型

- 并行：
  - IDE：133MB/s
  - SCSI：640MB/s
- 串口：
  - SATA：6Gbps
  - SAS：6Gbps
  - USB：480MB/s
- rpm: rotations per minute



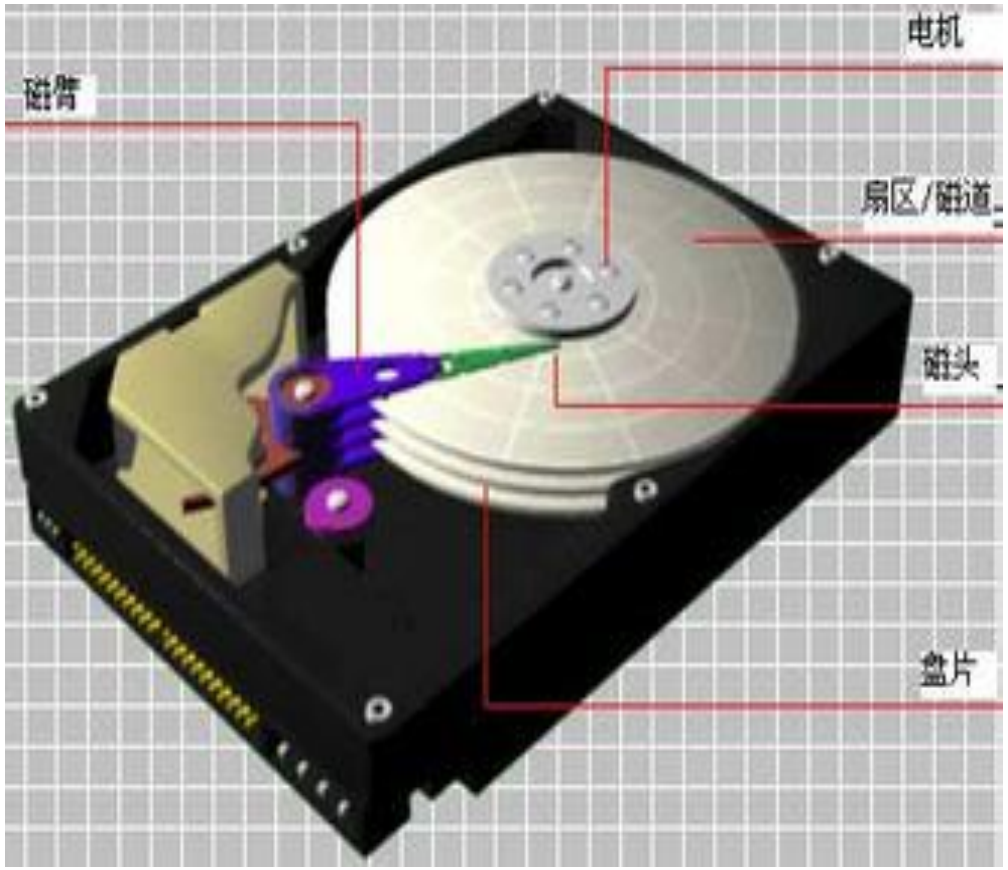
机械硬盘和固态硬盘

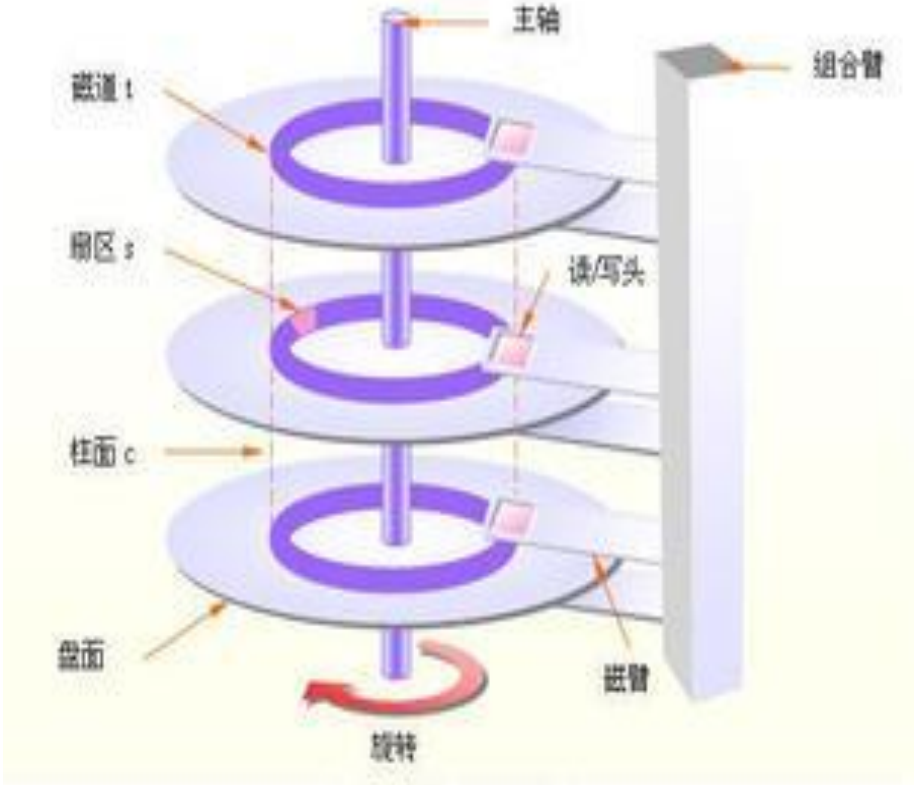
- 机械硬盘（HDD）：Hard Disk Drive，即是传统普通硬盘，主要由：盘片，磁头，盘片转轴及控制电机，磁头控制器，数据转换器，接口，缓存等几个部分组成。机械硬盘中所有的盘片都装在一个旋转轴上，每张盘片之间是平行的，在每个盘片的存储面上有一个磁头，磁头与盘片之间的距离比头发丝的直径还小，所有的磁头联在一个磁头控制器上，由磁头控制器负责各个磁头的运动。磁头可沿盘片的半径方向运动，加上盘片每分钟几千转的高速旋转，磁头就可以定位在盘片的指定位置上进行数据的读写操作。数据通过磁头由电磁流来改变极性方式被电磁流写到磁盘上，也可以通过相反方式读取。硬盘为精密设备，进入硬盘的空气必须过滤
- 固态硬盘（SSD）：Solid State Drive，用固态电子存储芯片阵列而制成的硬盘，由控制单元和存储单元（FLASH芯片、DRAM芯片）组成。固态硬盘在接口的规范和定义、功能及使用方法上与普通硬盘的完全相同，在产品外形和尺寸上也与普通硬盘一致
- 相较于HDD，SSD在防震抗摔、传输速率、功耗、重量、噪音上有明显优势，SSD传输速率性能是HDD的2倍
- 相较于SSD，HDD在价格、容量、使用寿命上占有绝对优势
- 硬盘有价，数据无价，目前SSD不能完全取代HHD

设备文件

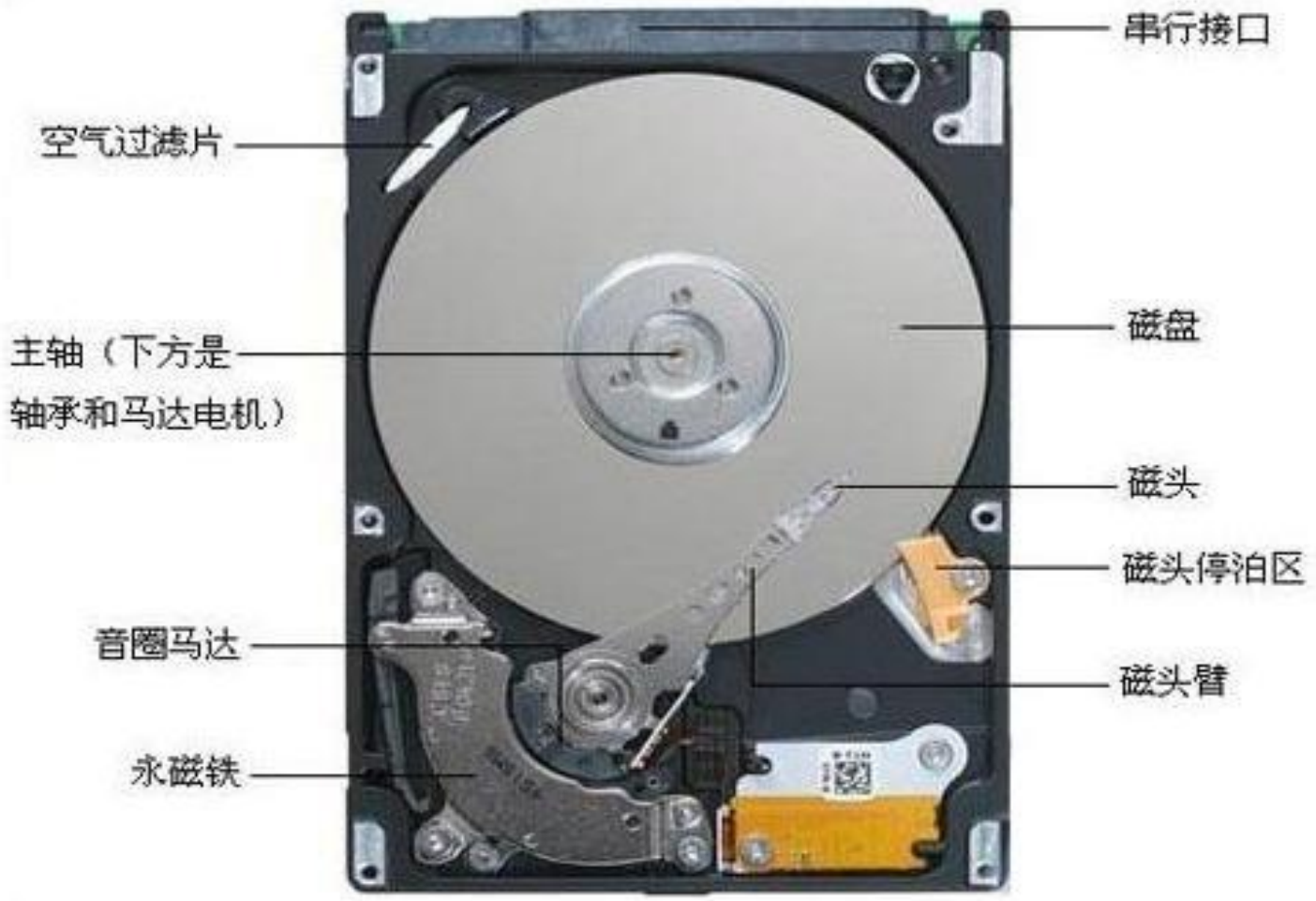
- 磁盘设备的设备文件命名：/dev/DEV_FILE
- SCSI, SATA, SAS, IDE,USB: /dev/sd
- 虚拟磁盘：/dev/vd
- 不同磁盘标识：a-z,aa,ab…
  /dev/sda, /dev/sdb, ...
- 同一设备上的不同分区：1,2, ...
  /dev/sda1, /dev/sda5
- 硬盘存储术语
  - head：磁头
  - track：磁道
  - cylinder: 柱面
  - sector: 扇区，512bytes

机械硬盘结构



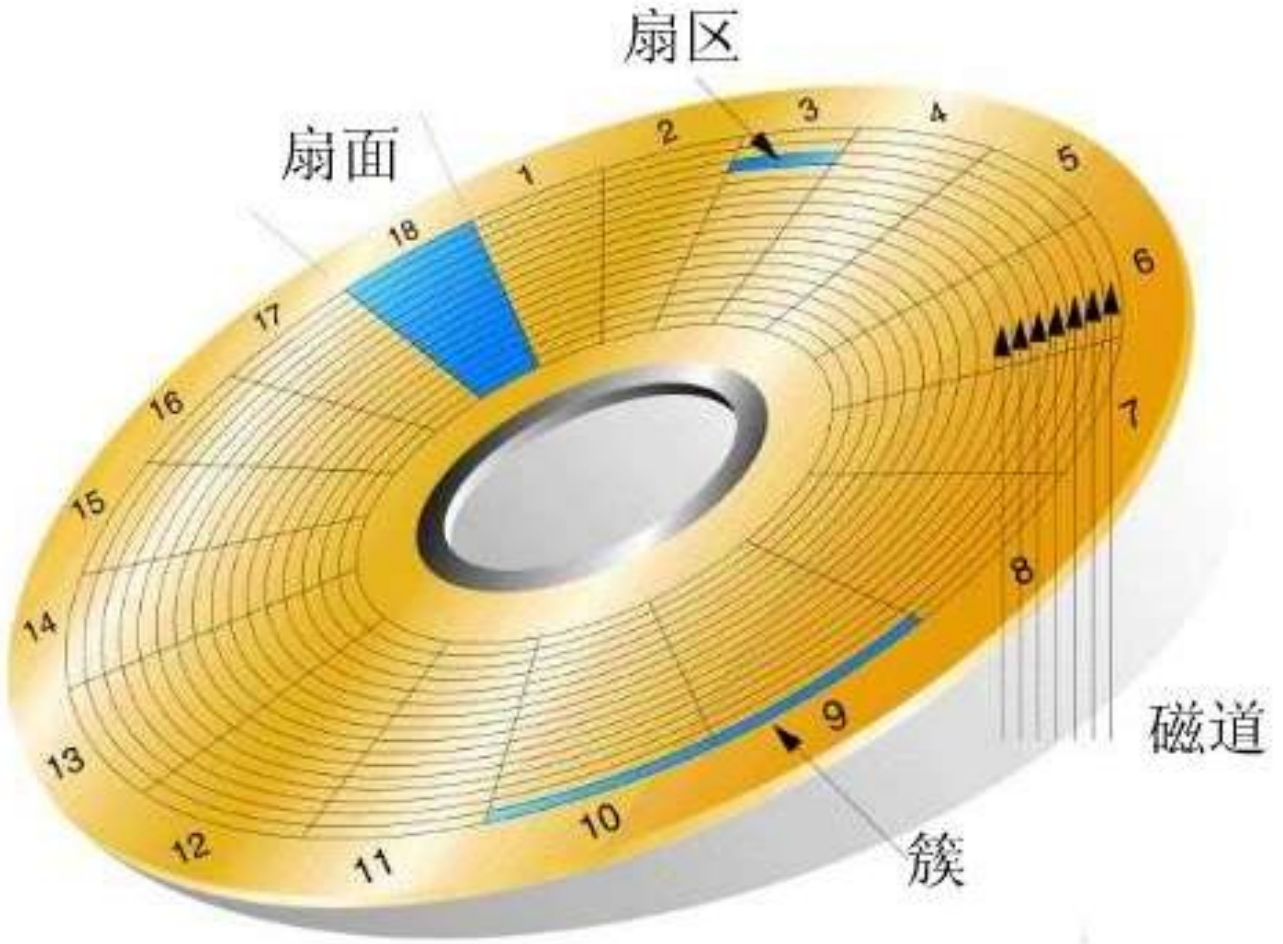


机械硬盘结构



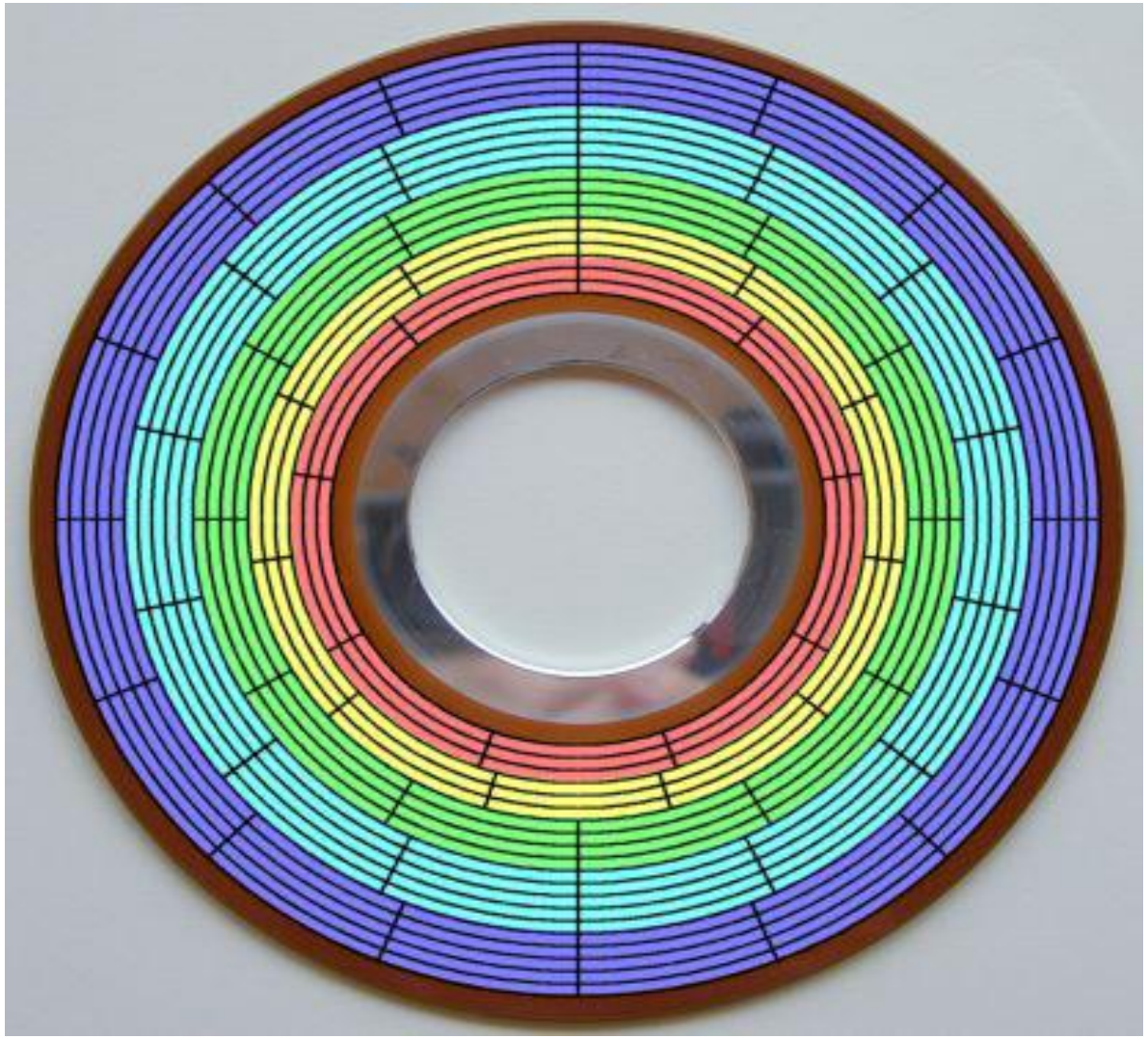
固态硬盘（SSD）和机械硬盘（HDD）

硬盘存储术语



区位记录磁盘扇区结构

- ZBR（Zoned Bit Recording）



CHS和LBA

- CHS
  - 采用24bit位寻址
  - 其中前10位表示cylinder，中间8位表示head，后面6位表示sector
  - 最大寻址空间8GB
- LBA（logical block addressing）
  - LBA是一个整数，通过转换成CHS格式完成磁盘具体寻址
  - ATA-1规范中定义了28位寻址模式，以每扇区512位组来计算，ATA-1所定义的28位LBA上限达到128 GiB。2002年ATA-6规范采用48位LBA，同样以每扇区512位组计算容量上限可达128 Petabytes
- 由于CHS寻址方式的寻址空间在大概8GB以内，所以在磁盘容量小于大概8GB时，可以使用CHS寻址方式或是LBA寻址方式；在磁盘容量大于大概8GB时，则只能使用LBA寻址方式

使用分区空间

- 设备识别
- 设备分区
- 创建文件系统
- 标记文件系统
- 在/etc/fstab文件中创建条目
- 挂载新的文件系统

磁盘分区

- 为什么分区
  - 优化I/O性能
  - 实现磁盘空间配额限制
  - 提高修复速度
  - 隔离系统和程序
  - 安装多个OS
  - 采用不同文件系统

分区

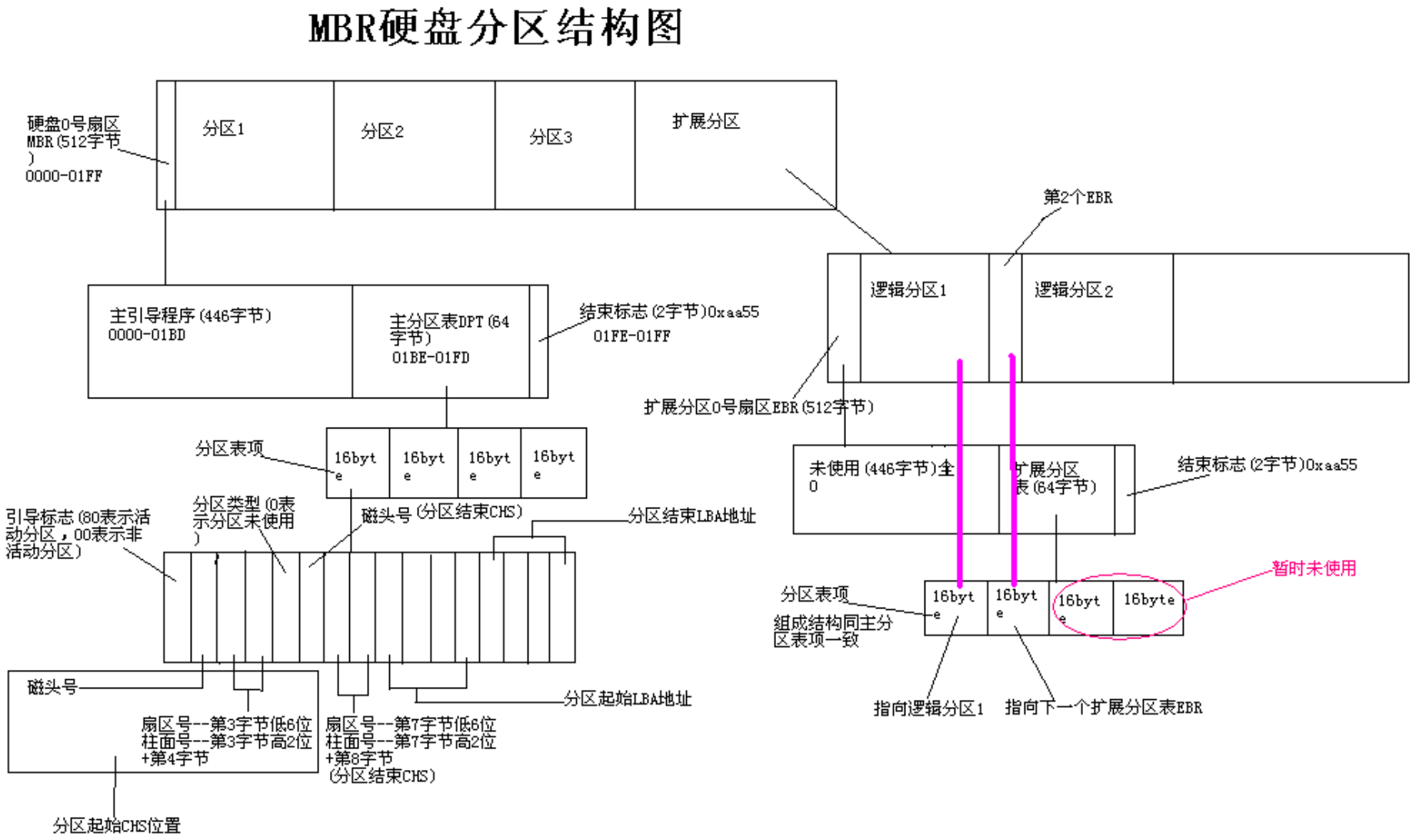
- 两种分区方式：MBR，GPT
- MBR: Master Boot Record，1982年，使用32位表示扇区数，分区不超过2T
- 如何分区：按柱面
- 0磁道0扇区：512bytes

  446bytes: boot loader

  64bytes：分区表，其中每16bytes标识一个分区

  2bytes: 55AA
- 4个主分区；3主分区+1扩展(N个逻辑分区)

MBR分区结构


MBR硬盘分区结构图
硬盘0号扇区
MBR(512字节)
0000-01FF
分区1
分区2
分区3
扩展分区
主引导程序(446字节)
0000-01BD
主分区表DPT(64字节)
01BE-01FD
结束标志(2字节)0xaa55
01FE-01FF
分区表项
16byte
16byte
16byte
16byte
引导标志(80表示活动分区，00表示非活动分区)
分区类型(0表示分区未使用)
磁头号(分区结束CHS)
分区结束LBA地址
磁头号
扇区号--第3字节低6位
柱面号--第3字节高2位
+第4字节
扇区号--第7字节低6位
柱面号--第7字节高2位
+第8字节
(分区结束CHS)
分区起始LBA地址
分区起始CHS位置
第2个EBR
逻辑分区1
逻辑分区2
扩展分区0号扇区EBR(512字节)
未使用(446字节)全0
扩展分区表(64字节)
结束标志(2字节)0xaa55
分区表项
组成结构同主分区表项一致
16byte
16byte
16byte
16byte
暂时未使用
指向逻辑分区1
指向下一个扩展分区表EBR

MBR分区结构

- 硬盘主引导记录MBR由4个部分组成
- 主引导程序（偏移地址0000H--0088H），它负责从活动分区中装载，并运行系统引导程序
- 出错信息数据区，偏移地址0089H--00E1H为出错信息，00E2H--01BDH全为0字节
- 分区表（DPT,Disk Partition Table）含4个分区项，偏移地址01BEH--01FDH,每个分区表项长16个字节，共64字节为分区项1、分区项2、分区项3、分区项4
- 结束标志字，偏移地址01FE--01FF的2个字节值为结束标志55AA

MBR结构

0000-0088	Master Boot Record 主引导程序	主引导 程序
0089-01BD	出错信息数据区	数据区
01BE-01CD	分区项1（16字节）	分区表
01CE-01DD	分区项2（16字节）	
01DE-01ED	分区项3（16字节）	
01EE-01FD	分区项4（16字节）	
01FE	55	结束标志
01FF	AA	

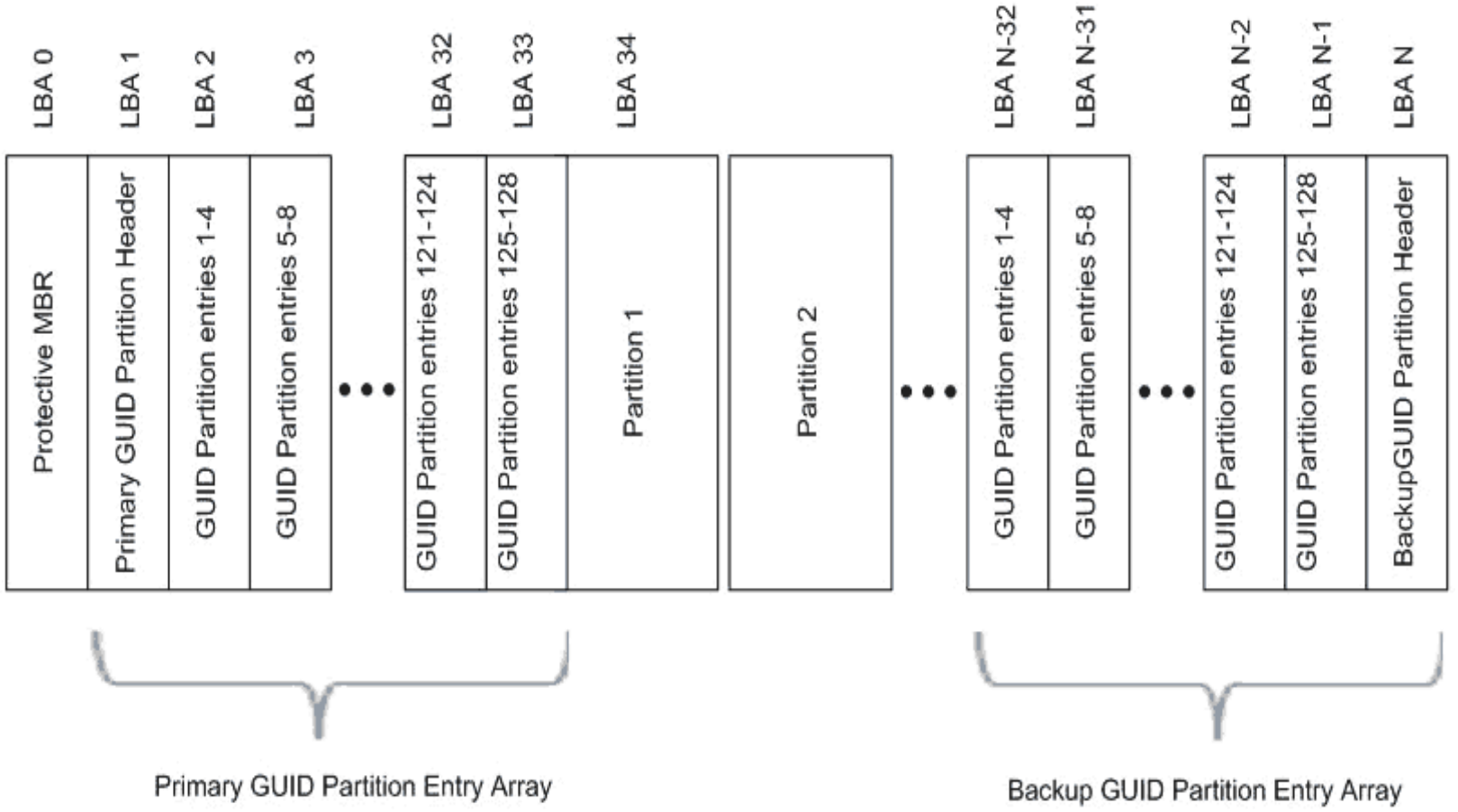
MBR中DPT结构

存贮字节位	内容及含义
第1字节	引导标志。若值为80H表示活动分区，若值为00H表示非活动分区。
第2、3、4字节	本分区的起始磁头号、扇区号、柱面号。其中： 磁头号——第2字节； 扇区号——第3字节的低6位； 柱面号——为第3字节高2位+第4字节8位。
第5字节	分区类型符。 00H——表示该分区未用（即没有指定）； 06H——FAT16基本分区； 0BH——FAT32基本分区； 05H——扩展分区； 07H——NTFS分区； 0FH——（LBA模式）扩展分区（83H为Linux分区等）。
第6、7、8字节	本分区的结束磁头号、扇区号、柱面号。其中： 磁头号——第6字节； 扇区号——第7字节的低6位； 柱面号——第7字节的高2位+第8字节。
第9、10、11、12字节	本分区之前已用了的扇区数。
第13、14、15、16字节	本分区的总扇区数。

GPT分区

- GPT:GUID（Globals Unique Identifiers） partition table 支持128个分区，使用64位，支持8Z（ 512Byte/block ）64Z （ 4096Byte/block)
- 使用128位UUID(Universally Unique Identifier) 表示磁盘和分区 GPT分区表自动备份在头和尾两份，并有CRC校验位
- UEFI (统一扩展固件接口)硬件支持GPT，使操作系统启动

GPT分区结构

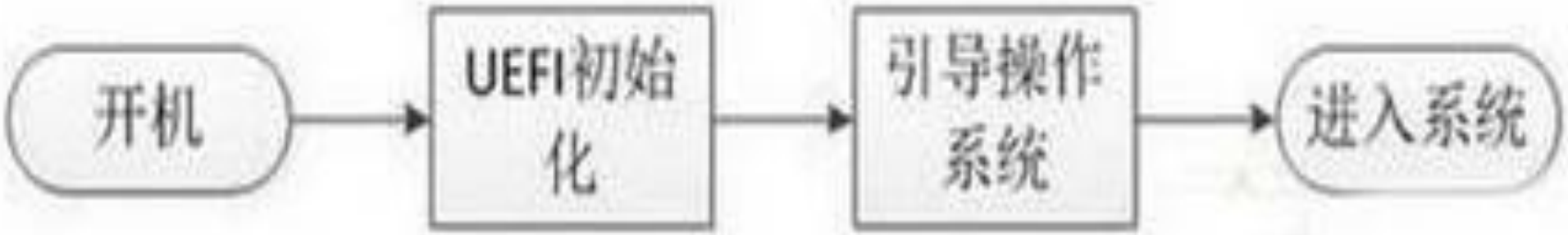


EFI部分又可以分为4个区域：EFI信息区(GPT头)、分区表、GPT分区、备份区域

BIOS+MBR与UEFI+GPT

传统BIOS运行流程

UEFI运行流程



管理分区

- 列出块设备
  - lsblk
- 创建分区使用：
  - fdisk 创建MBR分区
  - gdisk 创建GPT分区
  - parted 高级分区操作
- 重新设置内存中的内核分区表版本
  - partprobe

parted命令

- parted的操作都是实时生效的，小心使用
- 用法：parted [选项]... [设备 [命令 [参数]...]...]

  parted /dev/sdb mklabel gpt|msdos

  parted /dev/sdb print

  parted /dev/sdb mkpart primary 1 200 (默认M)

  parted /dev/sdb rm 1

  parted –l 列出分区信息

分区工具fdisk和gdisk

- gdisk /dev/sdb 类fdisk 的GPT分区工具
- fdisk -l [-u] [device...] 查看分区
- fdisk /dev/sdb 管理分区
- 子命令：
  - p　　分区列表
  - t　　更改分区类型
  - n　　创建新分区
  - d　　删除分区
  - v　　校验分区
  - u　　转换单位
  - w　　保存并退出
  - q　　不保存并退出

同步分区表

- 查看内核是否已经识别新的分区

  cat /proc/partations

- centos6通知内核重新读取硬盘分区表

  新增分区用

  partx -a /dev/DEVICE

  kpartx -a /dev/DEVICE -f: force

  删除分区用

  partx -d --nr M-N /dev/DEVICE

- CentOS 5，7: 使用partprobe

  partprobe [/dev/DEVICE]

文件系统

- 文件系统是操作系统用于明确存储设备或分区上的文件的方法和数据结构；即在存储设备上组织文件的方法。操作系统中负责管理和存储文件信息的软件结构称为文件管理系统，简称文件系统
- 从系统角度来看，文件系统是对文件存储设备的空间进行组织和分配，负责文件存储并对存入的文件进行保护和检索的系统。具体地说，它负责为用户建立文件，存入、读出、修改、转储文件，控制文件的存取，安全控制，日志，压缩，加密等
- 支持的文件系统：/lib/modules/`uname –r`/kernel/fs
- 各种文件系统：
  https://en.wikipedia.org/wiki/Comparison_of_file_systems

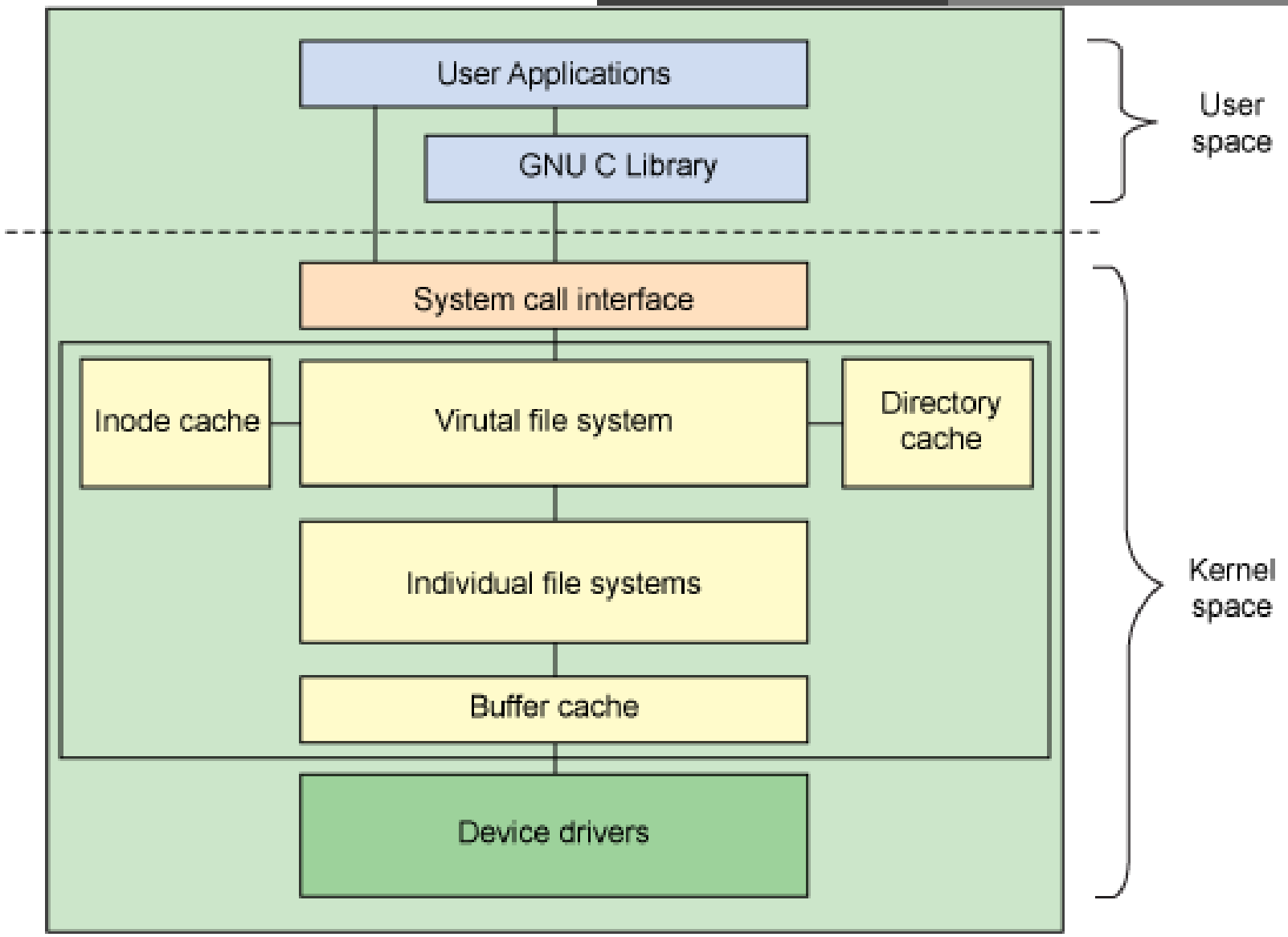
文件系统类型

- Linux文件系统：
  ext2(Extended file system) :适用于那些分区容量不是太大，更新也不频繁的情况，例如 /boot 分区
  ext3:是 ext2 的改进版本，其支持日志功能，能够帮助系统从非正常关机导致的异常中恢复。它通常被用作通用的文件系统
  ext4:是 ext 文件系统的最新版。提供了很多新的特性，包括纳秒级时间戳、创建和使用巨型文件(16TB)、最大1EB的文件系统，以及速度的提升
  xfs：SGI，支持最大8EB的文件系统
  btrfs（Oracle），reiserfs, jfs（AIX），swap
- 光盘：iso9660
- Windows：FAT32, exFAT,NTFS
- Unix: FFS（fast），UFS（unix），JFS2
- 网络文件系统：NFS, CIFS
- 集群文件系统：GFS2, OCFS2（oracle）
- 分布式文件系统： fastdfs,ceph, moosefs, mogilefs, glusterfs, Lustre
- RAW：未经处理或者未经格式化产生的文件系统

文件系统分类

- 根据其是否支持"journal"功能：
  - 日志型文件系统: ext3, ext4, xfs, ...
  - 非日志型文件系统: ext2, vfat
- 文件系统的组成部分：
  - 内核中的模块：ext4, xfs, vfat
  - 用户空间的管理工具：mkfs.ext4, mkfs.xfs,mkfs.vfat
- Linux的虚拟文件系统：VFS
- 查前支持的文件系统：cat /proc/filesystems

VFS


User Applications
GNU C Library
System call interface
Inode cache
Virutal file system
Directory cache
Individual file systems
Buffer cache
Device drivers
User space
Kernel space

文件系统选择

类型	支持限制	Root 分区	Boot分区	注释
单节点				
XFS	500TB	Yes	Yes	默认分区格式
Ext4	50TB	Yes	Yes	兼容 Ext3，Ext2
brtfs	50TB	Yes	Yes	技术预览
网络 / 多节点				
GFS2	2-16 个节点	Yes	No	集群文件共享存储

创建文件系统

- mkfs命令：
- (1) mkfs.FS_TYPE /dev/DEVICE
  - ext4
  - xfs
  - btrfs
  - vfat
- (2) mkfs -t FS_TYPE /dev/DEVICE
  - -L 'LABEL'　设定卷标

创建ext文件系统

- mke2fs：ext系列文件系统专用管理工具

  -t {ext2|ext3|ext4}　　指定文件系统类型

  -b {1024|2048|4096}　　指定块大小

  -L　'LABEL'　　设置卷标

  -j　　相当于 -t ext3

  mkfs.ext3 = mkfs -t ext3 = mke2fs -j = mke2fs -t ext3

  -i　#　为数据空间中每多少个字节创建一个inode；不应该小于block大小

  -N #　指定分区中创建多少个inode

  -I　　一个inode记录占用的磁盘空间大小，128---4096

  -m #　默认5%,为管理人员预留空间占总空间的百分比

  -O FEATURE[,...]　　启用指定特性

  -O ^FEATURE　　关闭指定特性

文件系统标签

- 指向设备的另一种方法
- 与设备无关
- blkid：块设备属性信息查看

  blkid [OPTION]... [DEVICE]

  -U UUID　根据指定的UUID来查找对应的设备

  -L LABEL　根据指定的LABEL来查找对应的设备
- e2label：管理ext系列文件系统的LABEL

  e2label DEVICE [LABEL]
- findfs ： 查找分区

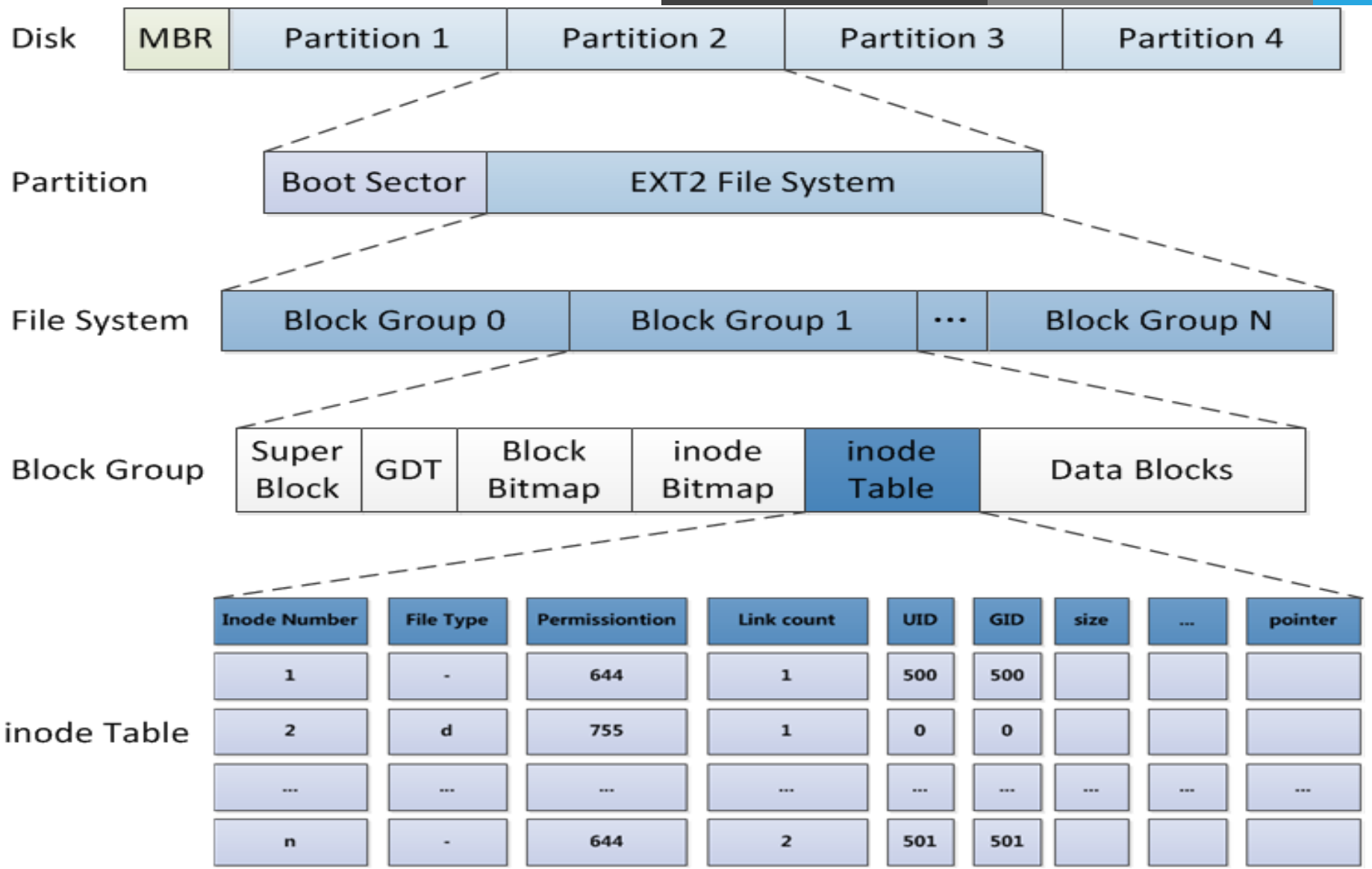
  findfs [options] LABEL=<label>

  findfs [options] UUID=<uuid>

tune2fs

- tune2fs：重新设定ext系列文件系统可调整参数的值
  - -l　　查看指定文件系统超级块信息；super block
  - -L 'LABEL ‘ 修改卷标
  - -m # 修预留给管理员的空间百分比
  - -j　　将ext2升级为ext3
  - -O　　文件系统属性启用或禁用, –O ^has_journal
  - -o　　调整文件系统的默认挂载选项，–o ^acl
  - -U UUID　　修改UUID号
- dumpe2fs：
  - 块分组管理，32768块
  - -h：查看超级块信息，不显示分组信息

超级块和INODE TABLE



文件系统检测和修复

- 常发生于死机或者非正常关机之后
- 挂载为文件系统标记为“no clean”
- 注意：一定不要在挂载状态下修复
- fsck: File System Check

  fsck.FS_TYPE

  fsck -t FS_TYPE

  -p　　自动修复错误

  -r　　交互式修复错误

  FS_TYPE　　一定要与分区上已经文件类型相同
- e2fsck：ext系列文件专用的检测修复工具

  -y　　自动回答为yes

  -f　　强制修复

挂载mount

- 挂载:将额外文件系统与根文件系统某现存的目录建立起关联关系，进而使得此目录做为其它文件访问入口的行为
- 卸载:为解除此关联关系的过程
- 把设备关联挂载点：mount Point

  mount
- 卸载时：可使用设备，也可以使用挂载点

  umount
- 挂载点下原有文件在挂载完成后会被临时隐藏
- 挂载点目录一般为空

用mount命令挂载文件系统

- 挂载方法：mount DEVICE MOUNT_POINT
- mount：通过查看/etc/mtab文件显示当前已挂载的所有设备
- mount [-fnrsvw] [-t vfstype] [-o options] device dir

  device：指明要挂载的设备；

  (1) 设备文件：例如/dev/sda5

  (2) 卷标：-L 'LABEL', 例如 -L 'MYDATA'

  (3) UUID, -U 'UUID'：例如 -U '0c50523c-43f1-45e7-85c0-a126711d406e'

  (4) 伪文件系统名称：proc, sysfs, devtmpfs, configfs

  dir：挂载点

  事先存在；建议使用空目录

  进程正在使用中的设备无法被卸载

mount常用命令选项

- -t vsftype　　指定要挂载的设备上的文件系统类型
- -r　　readonly，只读挂载
- -w　　read and write, 读写挂载
- -n　　不更新/etc/mtab，mount不可见
- -a　　自动挂载所有支持自动挂载的设备(定义在了/etc/fstab文件中，且挂载选项中有auto功能)
- -L 'LABEL'　　以卷标指定挂载设备
- -U 'UUID'　　以UUID指定要挂载的设备
- -B, --bind　　绑定目录到另一个目录上
- 查看内核追踪到的已挂载的所有设备

  cat /proc/mounts

mount常用命令选项

- -o options：(挂载文件系统的选项)，多个选项使用逗号分隔

async　异步模式	sync　同步模式,内存更改时，同时写磁盘
atime/noatime	包含目录和文件
diratime/nodiratime	目录的访问时间戳
auto/noauto	是否支持自动挂载,是否支持-a选项
exec/noexec	是否支持将文件系统上运行应用程序
dev/nodev	是否支持在此文件系统上使用设备文件
suid/nosuid	是否支持suid和sgid权限
remount	重新挂载
ro只读	rw读写
user/nouser	是否允许普通用户挂载此设备，/etc/fstab使用
acl	启用此文件系统上的acl功能
loop	使用loop设备

- defaults：相当于rw, suid, dev, exec, auto, nouser, async

卸载命令

- 查看挂载情况

  findmnt　MOUNT_POINT|device

- 查看正在访问指定文件系统的进程

  lsof MOUNT_POINT

  fuser -v MOUNT_POINT

- 终止所有在正访问指定的文件系统的进程

  fuser -km MOUNT_POINT

- 卸载

  umount DEVICE

  umount MOUNT_POINT

挂载点和/etc/fstab

- 配置文件系统体系
- 被mount、fsck和其它程序使用
- 系统重启时保留文件系统体系
- 可以在设备栏使用文件系统卷标
- 使用mount -a 命令挂载/etc/fstab中的所有文件系统

文件挂载配置文件

- /etc/fstab每行定义一个要挂载的文件系统
- 1、要挂载的设备或伪文件系统
  设备文件
  LABEL：LABEL=""
  UUID：UUID=""
  伪文件系统名称：proc, sysfs
- 2、挂载点
- 3、文件系统类型：ext4,xfs,nfs,none
- 4、挂载选项：defaults ，acl，bind
- 5、转储频率：0：不做备份　　1：每天转储　2：每隔一天转储
- 6、fsck检查的文件系统的顺序：允许的数字是0, 1, 和2
  0：不自检
  1：首先自检；一般只有rootfs才用
  2：非rootfs使用

处理交换文件和分区

- 交换分区是系统RAM的补充
- 基本设置包括：
  - 创建交换分区或者文件
  - 使用mkswap写入特殊签名
  - 在/etc/fstab文件中添加适当的条目
  - 使用swapon -a 激活交换空间

挂载交换分区

- 启用：swapon
  swapon [OPTION]... [DEVICE]
    -a：激活所有的交换分区
    -p PRIORITY：指定优先级
    /etc/fstab:pri=value
- 禁用：swapoff [OPTION]... [DEVICE]

SWAP的优先级

- 可以指定swap分区0到32767的优先级，值越大优先级越高
- 如果用户没有指定，那么核心会自动给swap指定一个优先级，这个优先级从-1开始，每加入一个新的没有用户指定优先级的swap，会给这个优先级减一
- 先添加的swap的缺省优先级比较高，除非用户自己指定一个优先级，而用户指定的优先级(是正数)永远高于核心缺省指定的优先级(是负数)
- 优化性能：分布存放，高性能磁盘存放

移动介质

- 挂载意味着使外来的文件系统看起来如同是主目录树的一部分
- 访问前、介质必须被挂载
- 摘除时，介质必须被卸载
- 按照默认设置，非根用户只能挂载某些设备（光盘、DVD、软盘、USB等等）
- 挂载点通常在/media 或/mnt下

使用光盘

- 在图形环境下自动启动挂载/run/media/<user>/<label>
- 否则就必须被手工挂载

  mount /dev/cdrom /mnt/
- eject命令卸载或弹出磁盘
- 创建ISO文件

  cp /dev/cdrom /root/centos7.iso

  mkisofs -r -o /root/etc.iso /etc
- 刻录光盘

  wodim –v –eject centos.iso

挂载USB介质

- 查看USB设备是否识别
  - lsusb
- 被内核探测为SCSI设备
  - /dev/sdaX、/dev/sdbX、或类似的设备文件
- 在图形环境中自动挂载
  - 图标在[计算机]窗口中创建
  - 挂载在/run/media/<user>/<label>
- 手动挂载
  - mount /dev/sdb1 /mnt

常见工具

- 文件系统空间占用等信息的查看工具：

  df [OPTION]... [FILE]...

  -H 以1000为单位

  -T 文件系统类型

  -h: human-readable

  -i：inodes instead of blocks

  -P: 以Posix兼容的格式输出

- 查看某目录总体空间占用状态：

  du [OPTION]... DIR

  -h: human-readable

  -s: summary --max-depth

工具dd

- dd命令：convert and copy a file
- 用法：

dd if=/PATH/FROM/SRC of=/PATH/TO/DEST

bs=#：block size, 复制单元大小

count=#：复制多少个bs

of=file	写到所命名的文件而不是到标准输出
if=file	从所命名文件读取而不是从标准输入
bs=size	指定块大小（既是是ibs也是obs)
ibs=size	一次读size个byte
obs=size	一次写size个byte
cbs=size	一次转化size个byte
skip=blocks	从开头忽略blocks个ibs大小的块
seek=blocks	从开头忽略blocks个obs大小的块
count=n	只拷贝n个记录

工具dd

- conv=conversion[,conversion...] 用指定的参数转换文件
- 转换参数:
- ascii 转换 EBCDIC 为 ASCII
- ebcdic 转换 ASCII 为 EBCDIC
- lcase 把大写字符转换为小写字符
- ucase 把小写字符转换为大写字符
- nocreat 不创建输出文件
- noerror 出错时不停止
- notrunc 不截短输出文件
- sync 把每个输入块填充到ibs个字节，不足部分用空(NUL)字符补齐
- Fdatasync 写完成前，物理写入输出文件

工具dd

- 备份MBR

  dd if=/dev/sda of=/tmp/mbr.bak bs=512 count=1

- 破坏MBR中的bootloader

  dd if=/dev/zero of=/dev/sda bs=64 count=1 seek=446

- 有一个大与2K的二进制文件fileA。现在想从第64个字节位置开始读取，需要读取的大小是128Byts。又有fileB, 想把上面读取到的128Bytes写到第32个字节开始的位置，替换128Bytes，实现如下

  dd if=fileA of=fileB bs=1 count=128 skip=63 seek=31 conv=notrunc

工具dd

- 备份：
  dd if=/dev/sdx of=/dev/sdy
  将本地的/dev/sdx整盘备份到/dev/sdy
  dd if=/dev/sdx of=/path/to/image
  将/dev/sdx全盘数据备份到指定路径的image文件
  dd if=/dev/sdx | gzip >/path/to/image.gz
  备份/dev/sdx全盘数据，并利用gzip压缩，保存到指定路径
- 恢复：
  dd if=/path/to/image of=/dev/sdx
  将备份文件恢复到指定盘
  gzip -dc /path/to/image.gz | dd of=/dev/sdx
  将压缩的备份文件恢复到指定盘

工具dd

- 拷贝内存资料到硬盘

  dd if=/dev/mem of=/root/mem.bin bs=1024

将内存里的数据拷贝到root目录下的mem.bin文件

- 从光盘拷贝iso镜像

  dd if=/dev/cdrom of=/root/cd.iso

拷贝光盘数据到root文件夹下，并保存为cd.iso文件

- 销毁磁盘数据

  dd if=/dev/urandom of=/dev/sda1

利用随机的数据填充硬盘，在某些必要的场合可以用来销毁数据，执行此操作以后，/dev/sda1将无法挂载，创建和拷贝操作无法执行

工具dd

- 得到最恰当的block size

```
dd if=/dev/zero of=/root/1Gb.file bs=1024 count=1000000
dd if=/dev/zero of=/root/1Gb.file bs=2048 count=500000
dd if=/dev/zero of=/root/1Gb.file bs=4096 count=250000
```

通过比较dd指令输出中命令的执行时间，即可确定系统最佳的block size大小

工具dd

- 测试硬盘写速度

```
dd if=/dev/zero of=/root/1Gb.file bs=1024 count=1000000
```

- 测试硬盘读速度

```
dd if=/root/1Gb.file bs=64k | dd of=/dev/null
```

工具dd

- 修复硬盘

dd if=/dev/sda of=/dev/sda

当硬盘较长时间（比如1,2年）放置不使用后，磁盘上会产生消磁点。当磁头读到这些区域时会遇到困难，并可能导致I/O错误。当这种情况影响到硬盘的第一个扇区时，可能导致硬盘报废。上边的命令有可能使这些数据起死回生,且这个过程是安全高效的

练习

- 1、创建一个2G的文件系统，块大小为2048byte，预留1%可用空间,文件系统ext4，卷标为TEST，要求此分区开机后自动挂载至/test目录，且默认有acl挂载选项
- 2、写一个脚本，完成如下功能：

  (1) 列出当前系统识别到的所有磁盘设备

  (2) 如磁盘数量为1，则显示其空间使用信息

  否则，则显示最后一个磁盘上的空间使用信息
- 3、将CentOS6的CentOS-6.10-x86_64-bin-DVD1.iso和CentOS-6.10-x86_64-bin-DVD2.iso两个文件，合并成一个CentOS-6.10-x86_64-Everything.iso文件，并将其配置为yum源

什么是RAID

- RAID:Redundant Arrays of Inexpensive（Independent） Disks
- 1988年由加利福尼亚大学伯克利分校（University of California-Berkeley） “A Case for Redundant Arrays of Inexpensive Disks”
- 多个磁盘合成一个“阵列”来提供更好的性能、冗余，或者两者都提供

RAID

- 提高IO能力
  - 磁盘并行读写
- 提高耐用性
  - 磁盘冗余来实现
- 级别：多块磁盘组织在一起的工作方式有所不同
- RAID实现的方式
  - 外接式磁盘阵列：通过扩展卡提供适配能力
  - 内接式RAID：主板集成RAID控制器，安装OS前在BIOS里配置
  - 软件RAID：通过OS实现

RAID级别

- RAID-0：条带卷，strip
- RAID-1：镜像卷，mirror
- RAID-2
- ..
- RAID-5
- RAID-6
- RAID-10
- RAID-01

RAID级别

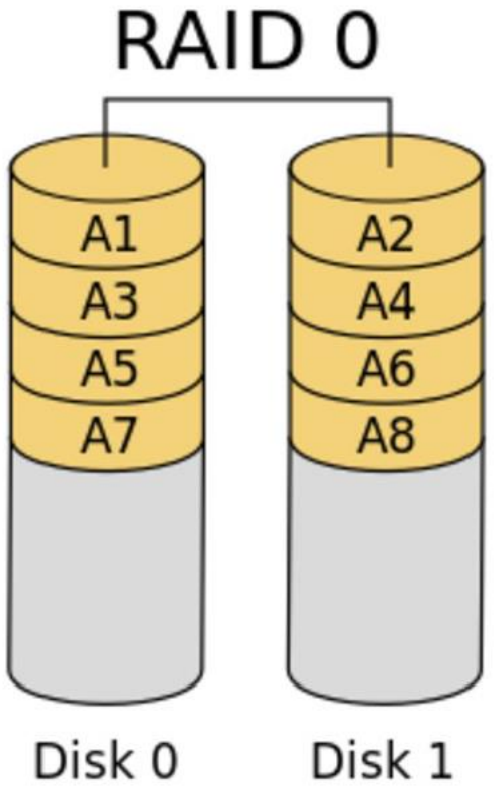
- RAID-0：

  读、写性能提升

  可用空间：N*min(S1,S2,...)

  无容错能力

  最少磁盘数：2, 2+



RAID级别

- RAID-1：

  读性能提升、写性能略有下降

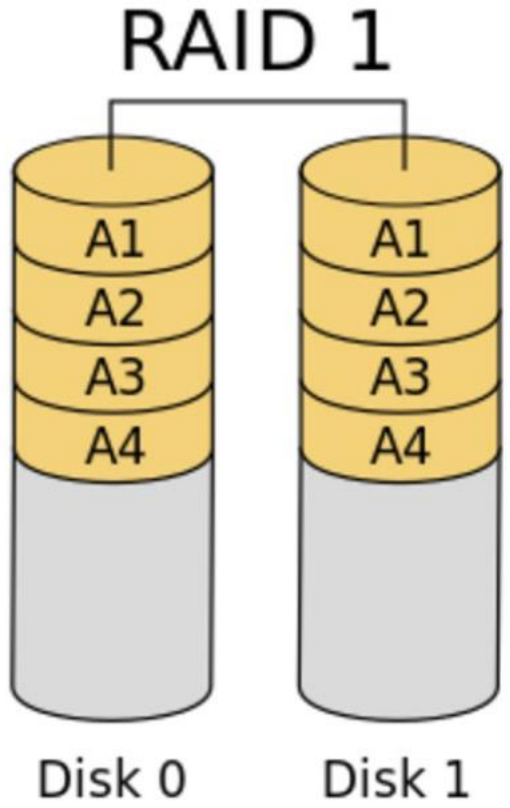
  可用空间：1*min(S1,S2,...)

  有冗余能力

  最少磁盘数：2, 2N

- RAID-4：

  多块数据盘异或运算值存于专用校验盘



RAID级别

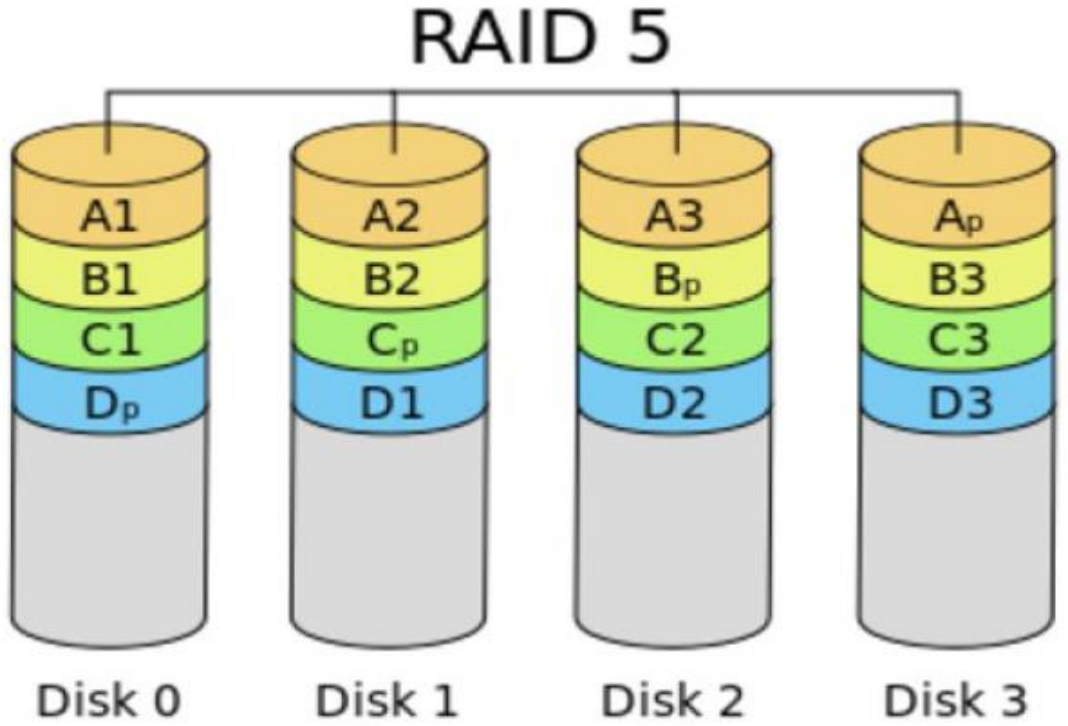
- RAID-5：

  读、写性能提升

  可用空间：(N-1)*min(S1,S2,...)

  有容错能力：允许最多1块磁盘损坏

  最少磁盘数：3, 3+



RAID级别

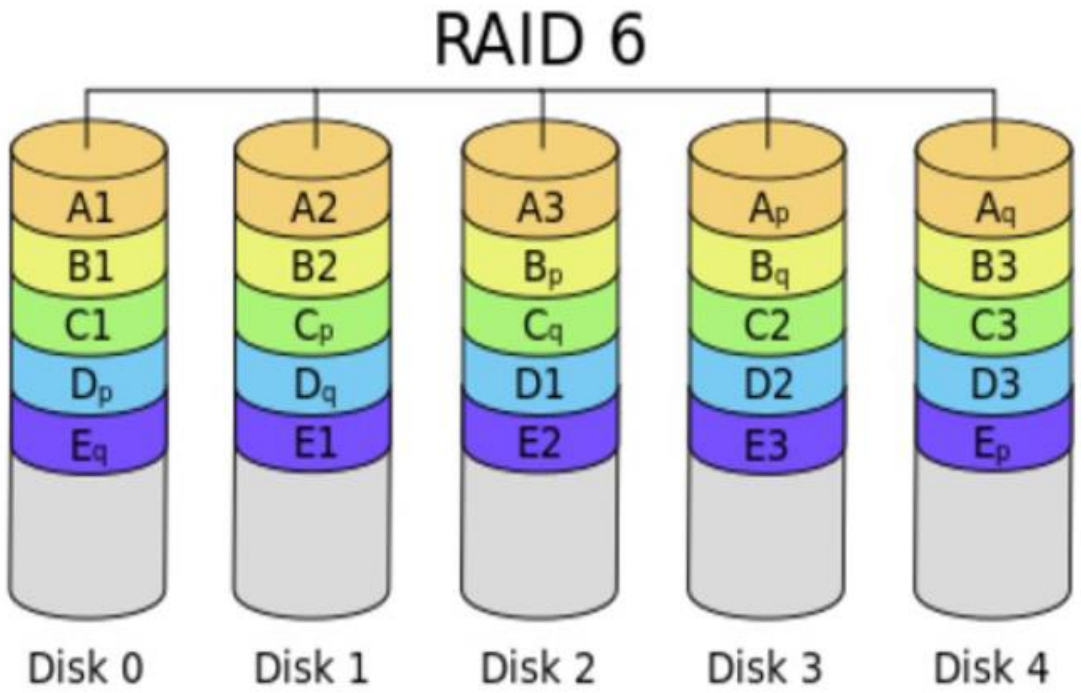
- RAID-6：

  读、写性能提升

  可用空间：(N-2)*min(S1,S2,...)

  有容错能力：允许最多2块磁盘损坏

  最少磁盘数：4, 4+



RAID级别

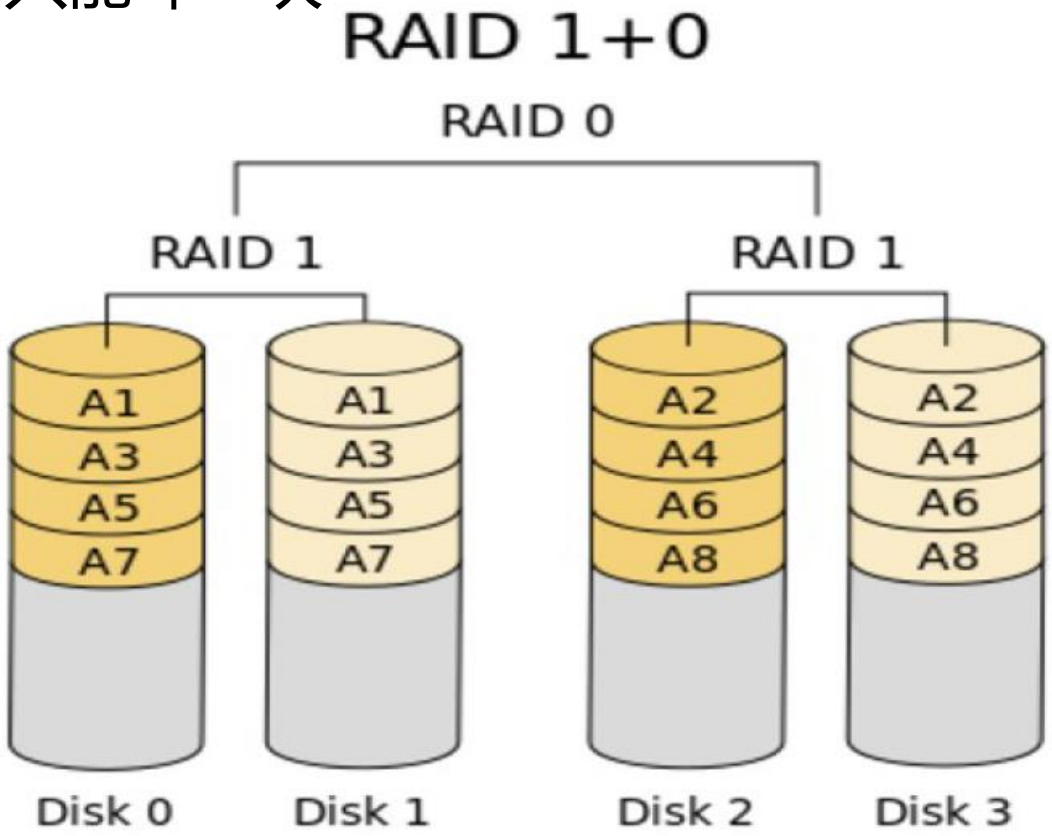
- RAID-10：

  读、写性能提升

  可用空间：N*min(S1,S2,...)/2

  有容错能力：每组镜像最多只能坏一块

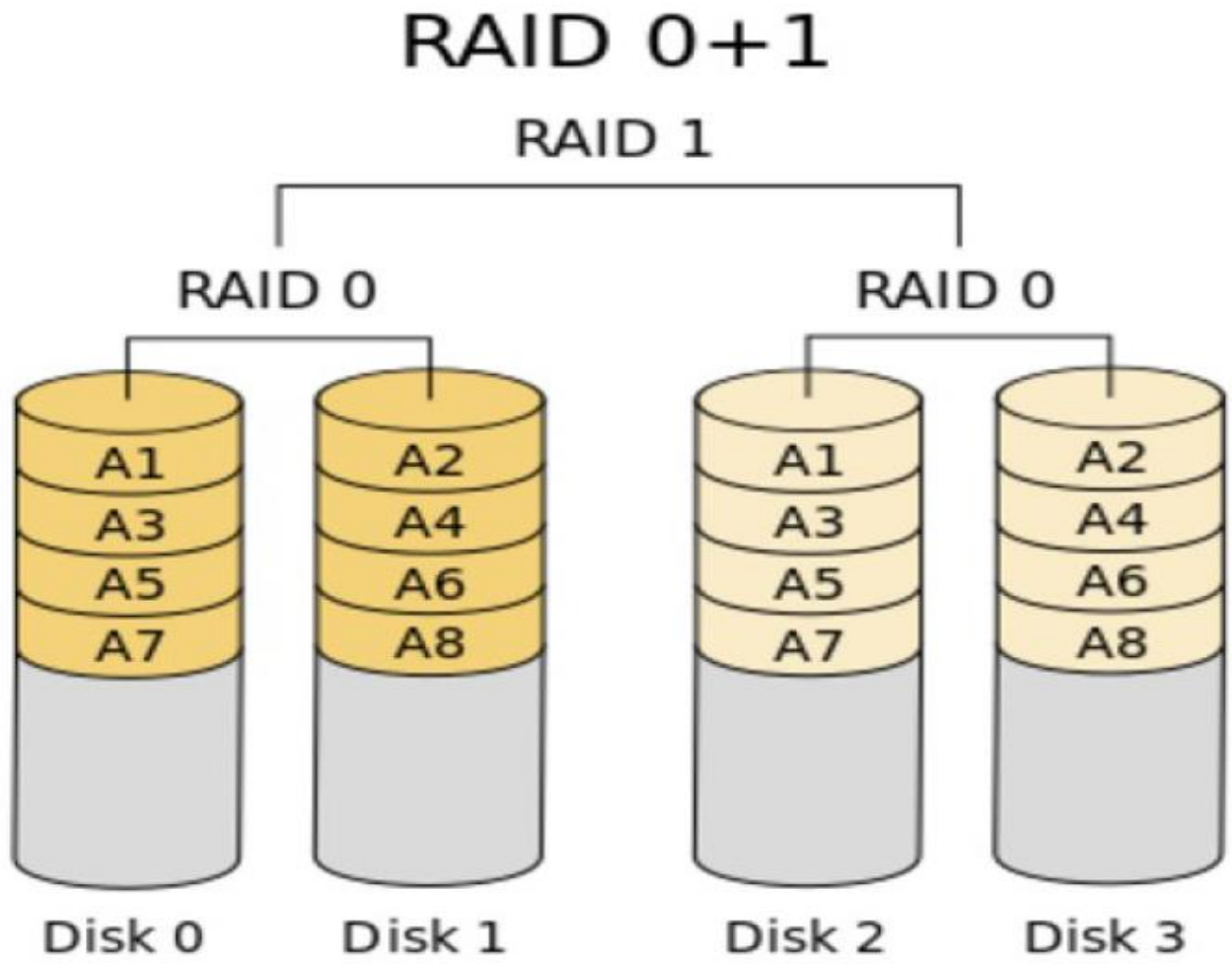
  最少磁盘数：4, 4+



RAID级别

- RAID-01

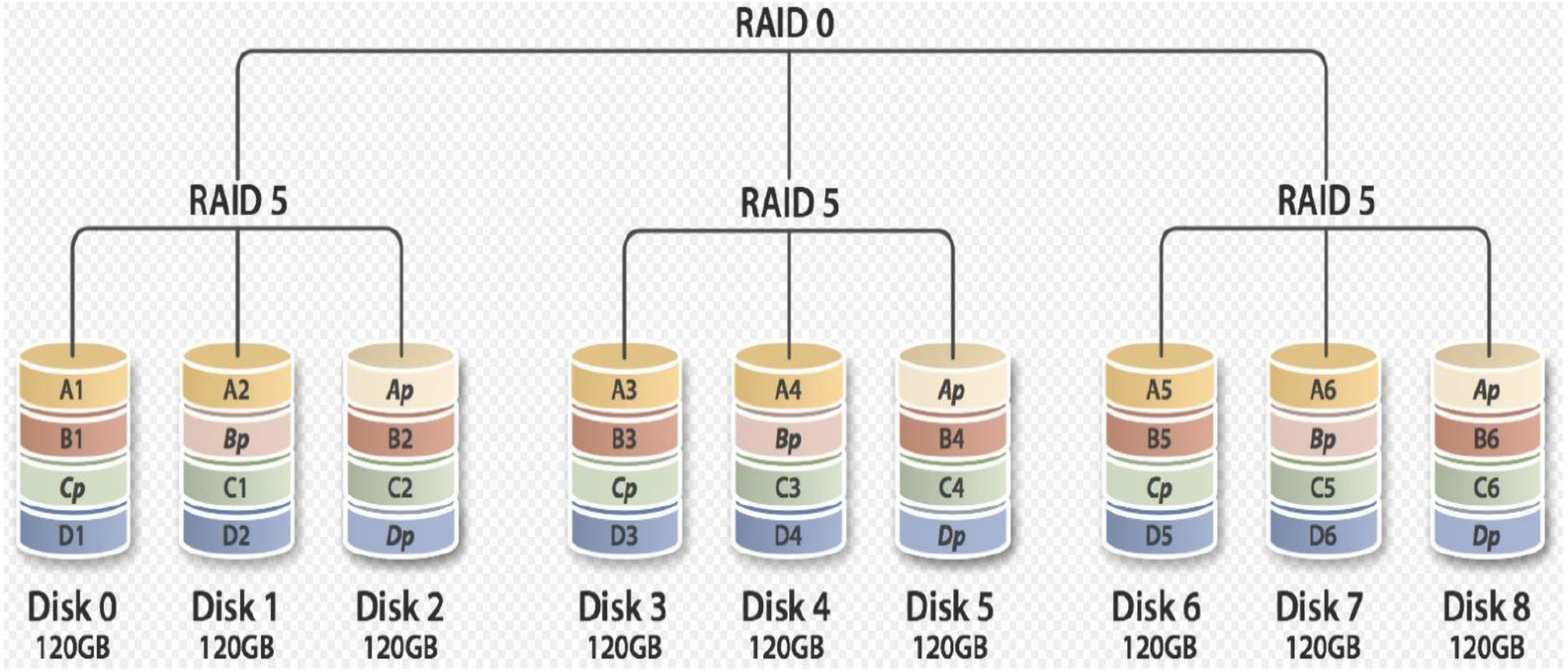
  多块磁盘先实现RAID0,再组合成RAID1



RAID级别

- RAID-50

  多块磁盘先实现RAID5,再组合成RAID0

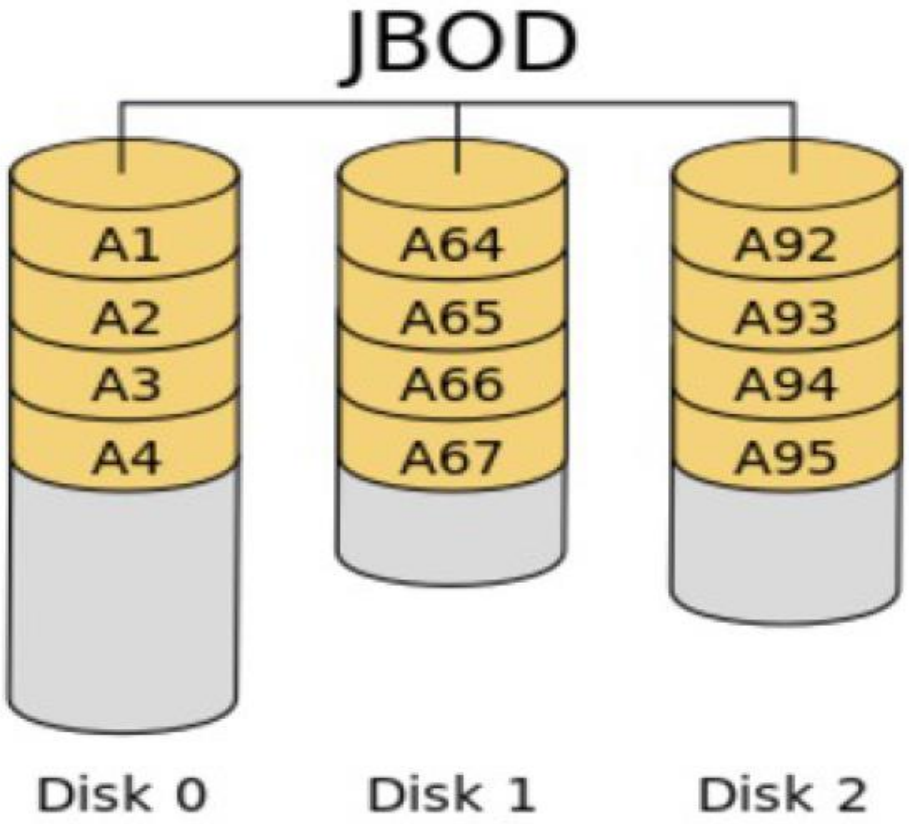


RAID级别

- JBOD：Just a Bunch Of Disks

  功能：将多块磁盘的空间合并一个大的连续空间使用

  可用空间：sum(S1,S2,...)



RAID级别

- RAID7

  可以理解为一个独立存储计算机，自身带有操作系统和管理工具，可以独立运行，理论上性能最高的RAID模式

- 常用级别：

  RAID-0, RAID-1, RAID-5, RAID-10, RAID-50, JBOD

软RAID

- mdadm：为软RAID提供管理界面
- 为空余磁盘添加冗余
- 结合内核中的md(multi devices)
- RAID设备可命名为/dev/md0、/dev/md1、/dev/md2、/dev/md3等

软件RAID的实现

- mdadm：模式化的工具
- 命令的语法格式：mdadm [mode] <raiddevice> [options] <component-devices>
- 支持的RAID级别：LINEAR, RAID0, RAID1, RAID4, RAID5, RAID6, RAID10
- 模式：
  - 创建：-C
  - 装配：-A
  - 监控：-F
  - 管理：-f, -r, -a
- <raiddevice>: /dev/md#
- <component-devices>: 任意块设备

软件RAID的实现

- -C: 创建模式
  - -n #: 使用#个块设备来创建此RAID
  - -l #：指明要创建的RAID的级别
  - -a {yes|no}：自动创建目标RAID设备的设备文件
  - -c CHUNK_SIZE: 指明块大小,单位k
  - -x #: 指明空闲盘的个数
- -D：显示raid的详细信息

  mdadm -D /dev/md#
- 管理模式：
  - -f: 标记指定磁盘为损坏
  - -a: 添加磁盘
  - -r: 移除磁盘
- 观察md的状态： cat /proc/mdstat

软RAID配置示例

- 使用mdadm创建并定义RAID设备

  mdadm -C /dev/md0 -a yes -l 5 -n 3 -x 1 /dev/sd{b,c,d,e}1

- 用文件系统对每个RAID设备进行格式化

  mkfs.xfs /dev/md0

- 测试RAID设备
- 使用mdadm检查RAID设备的状况

  mdadm --detail|D /dev/md0

- 增加新的成员

  mdadm –G /dev/md0 –n4 -a /dev/sdf1

软RAID测试和修复

- 模拟磁盘故障

  mdadm /dev/md0 -f /dev/sda1

- 移除磁盘

  mdadm /dev/md0 –r /dev/sda1

- 从软件RAID磁盘修复磁盘故障
  - 替换出故障的磁盘然后开机
  - 在备用驱动器上重建分区
  - mdadm /dev/md0 -a /dev/sda1
- mdadm、/proc/mdstat及系统日志信息

软RAID管理

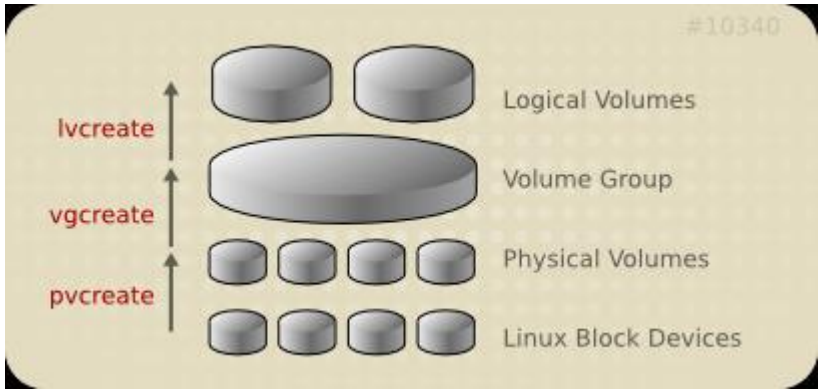
- 生成配置文件：mdadm –D –s >> /etc/mdadm.conf
- 停止设备：mdadm –S /dev/md0
- 激活设备：mdadm –A –s /dev/md0 激活
- 强制启动：mdadm –R /dev/md0
- 删除raid信息：mdadm --zero-superblock /dev/sdb1

练习

- 1：创建一个可用空间为1G的RAID1设备，文件系统为ext4，有一个空闲盘，开机可自动挂载至/backup目录
- 2：创建由三块硬盘组成的可用空间为2G的RAID5设备，要求其chunk大小为256k，文件系统为ext4，开机可自动挂载至/mydata目录

逻辑卷管理器（LVM）

- 允许对卷进行方便操作的抽象层，包括重新设定文件系统的大小
- 允许在多个物理设备间重新组织文件系统
  - 将设备指定为物理卷
  - 用一个或者多个物理卷来创建一个卷组
  - 物理卷是用固定大小的物理区域（Physical Extent，PE）来定义的
  - 在物理卷上创建的逻辑卷是由物理区域（PE）组成
  - 可以在逻辑卷上创建文件系统



LVM介绍

- LVM: Logical Volume Manager， Version: 2
- dm: device mapper：将一个或多个底层块设备组织成一个逻辑设备的模块
- 设备名：/dev/dm-#
- 软链接：

    /dev/mapper/VG_NAME-LV_NAME

        /dev/mapper/vol0-root

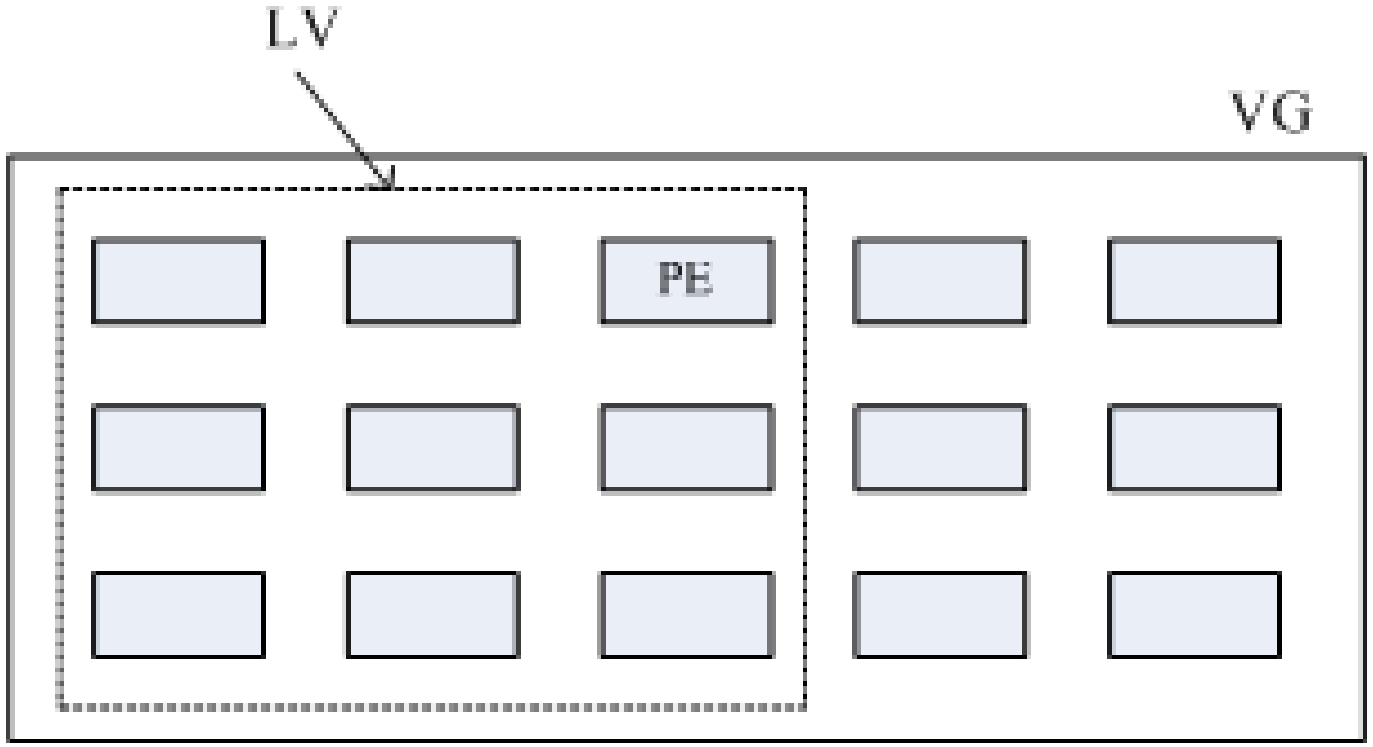
    /dev/VG_NAME/LV_NAME

        /dev/vol0/root

LVM更改文件系统的容量

- LVM可以弹性的更改LVM的容量

通过交换PE来进行资料的转换，将原来LV内的PE转移到其他的设备中以降低LV的容量，或将其他设备中的PE加到LV中以加大容量



pv管理工具

- 显示pv信息

  pvs：简要pv信息显示

  pvdisplay
- 创建pv

  pvcreate /dev/DEVICE
- 删除pv

  pvremove /dev/DEVICE

vg管理工具

- 显示卷组

  vgs

  vgdisplay

- 创建卷组

  vgcreate [-s #[kKmMgGtTpPeE]] VolumeGroupName PhysicalDevicePath [PhysicalDevicePath...]

- 管理卷组

  vgextend VolumeGroupName PhysicalDevicePath [PhysicalDevicePath...]

  vgreduce VolumeGroupName PhysicalDevicePath [PhysicalDevicePath...]

- 删除卷组

  先做pvmove，再做vgremove

lv管理工具

- 显示逻辑卷

  lvs

  Lvdisplay

- 创建逻辑卷

  lvcreate -L #[mMgGtT] -n NAME VolumeGroup

  lvcreate -l 60%VG -n mylv testvg

  lvcreate -l 100%FREE -n yourlv testvg

- 删除逻辑卷

  lvremove /dev/VG_NAME/LV_NAME

- 重设文件系统大小

  fsadm [options] resize device [new_size[BKMGTEP]]

  resize2fs [-f] [-F] [-M] [-P] [-p] device [new_size]

  xfs_growfs /mountpoint

扩展和缩减逻辑卷

- 扩展逻辑卷：

```
lvextend -L [+]#[mMgGtT] /dev/VG_NAME/LV_NAME
resize2fs /dev/VG_NAME/LV_NAME
lvresize -r -l +100%FREE /dev/VG_NAME/LV_NAME
```

- 缩减逻辑卷：

```
umount /dev/VG_NAME/LV_NAME
e2fsck -f /dev/VG_NAME/LV_NAME
resize2fs /dev/VG_NAME/LV_NAME #[mMgGtT]
lvreduce -L [-]#[mMgGtT] /dev/VG_NAME/LV_NAME
mount
```

跨主机迁移卷组

- 源计算机上
  - 1 在旧系统中，umount所有卷组上的逻辑卷
  - 2 禁用卷组
    vgchange –a n vg0
    lvdisplay
  - 3 导出卷组
    vgexport vg0
    pvscan
    vgdisplay
    拆下旧硬盘
- 在目标计算机上
  - 4 在新系统中安装旧硬盘，并导入卷组：vgimport vg0
  - 5 vgchange –ay vg0 启用
  - 6 mount所有卷组上的逻辑卷

创建逻辑卷示例

- 创建物理卷

  pvcreate /dev/sda3

- 为卷组分配物理卷

  vgcreate vg0 /dev/sda3

- 从卷组创建逻辑卷

  lvcreate -L 256M -n data vg0

  mkfs.xfs -j /dev/vg0/data

- 挂载

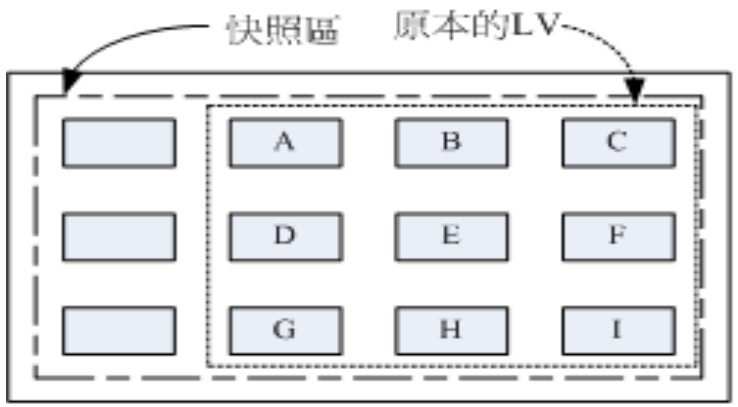
  mount /dev/vg0/data /mnt/data

逻辑卷管理器快照

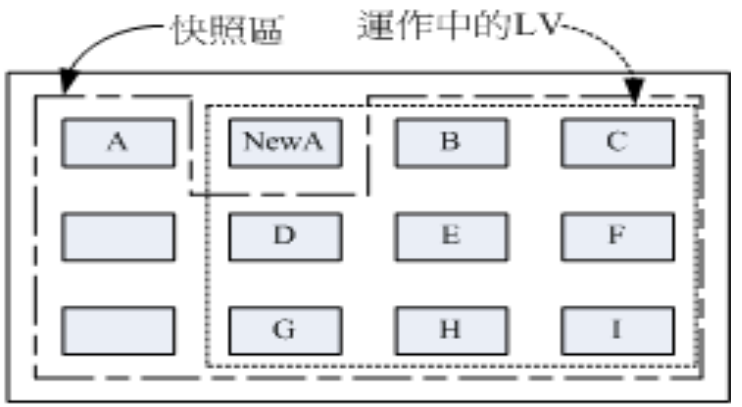
- 快照是特殊的逻辑卷，它是在生成快照时存在的逻辑卷的准确拷贝
- 对于需要备份或者复制的现有数据临时拷贝以及其它操作来说，快照是最合适的选择
- 快照只有在它们和原来的逻辑卷不同时才会消耗空间
  - 在生成快照时会分配给它一定的空间，但只有在原来的逻辑卷或者快照有所改变才会使用这些空间
  - 当原来的逻辑卷中有所改变时，会将旧的数据复制到快照中。
  - 快照中只含有原来的逻辑卷中更改的数据或者自生成快照后的快照中更改的数据
  - 建立快照的卷大小只需要原始逻辑卷的15%～20%就够了,也可以使用lvextend放大快照

逻辑卷管理器快照

- 快照就是将当时的系统信息记录下来，就好像照相一般，若将来有任何数据改动了，则原始数据会被移动到快照区，没有改动的区域则由快照区和文件系统共享



此時的A～I的PE爲共用區域



A更動過，快照區保留舊A，未更動的B～I部分的PE爲共用區

- 由于快照区与原本的LV共用很多PE的区块，因此快照与被快照的LV必须在同一个VG中.系统恢复的时候的文件数量不能高于快照区的实际容量

使用LVM快照

- 为现有逻辑卷创建快照

  lvcreate -l 64 -s -n data-snapshot -p r /dev/vg0/data

- 挂载快照

  mkdir -p /mnt/snap

  mount -o ro /dev/vg0/data-snapshot   /mnt/snap

- 恢复快照

  umount /dev/vg0/data-snapshot

  umount /dev/vg0/data

  lvconvert --merge /dev/vg0/data-snapshot

- 删除快照

  umount /mnt/databackup

  lvremove /dev/vg0/databackup

练习

- 1、创建一个至少有两个PV组成的大小为20G的名为testvg的VG；要求PE大小为16MB, 而后在卷组中创建大小为5G的逻辑卷testlv；挂载至/users目录
- 2、 新建用户archlinux，要求其家目录为/users/archlinux，而后su切换至archlinux用户，复制/etc/pam.d目录至自己的家目录
- 3、扩展testlv至7G，要求archlinux用户的文件不能丢失
- 4、收缩testlv至3G，要求archlinux用户的文件不能丢失
- 5、对testlv创建快照，并尝试基于快照备份数据，验证快照的功能

关于马哥教育

马哥教育
IT人的高薪职业学院

- 博客：http://mageedu.blog.51cto.com
- 主页：http://www.magedu.com
- QQ：1661815153, 113228115
- QQ群：203585050, 279599283

祝大家学业有成

谢　谢