任务背景

某天接到短信报警提示，显示某主机的根分区空间使用率超过85%，该主机用于影评(mysql)和报表数据库(oracle)。经查看发现其中MySQL数据库的数据文件存放在/usr/local/mysql/中，占用根文件系统空间导致。由于前期规划不合理，没有将业务数据和系统数据分开。经研究决定，要将影评的数据库单独放到另一块磁盘上，并且实现逻辑卷管理。

任务要求

1. 保证数据库完整的情况下将影评数据库迁移到另外一块新添加的磁盘上
2. 考虑到数据增长情况，新磁盘使用lvm逻辑卷管理，方便日后动态扩容

任务拆解

1. 需要有一块可用硬盘（需要在虚拟机里增加一块硬盘）
2. 使用lvm方式管理磁盘（学习lvm相关的知识点）
3. 迁移数据库（MySQL）业务维护时间（23:00—）
    1. 停监控（根据情况）
    2. 停前端应用
    3. 停MySQL数据库
    4. 备份数据库（全备）
    5. 将原有的数据库同步到新的设备存储上（数据同步rsync）
    6. 启动数据库（保证没有问题）
    7. 启动前端应用
    8. 测试人员测试
    9. 开门营业（记得打开监控）

涉及知识点

- 磁盘分区相关概念和工具(fdisk)
- LVM逻辑卷相关概念和工具使用（重点）

课程目标

- 能够使用fdisk命令对磁盘进行分区
- 熟悉设备的挂载方式（手动、开机自动、autofs自动）
- 理解物理卷、卷组、逻辑卷的概念
- 能够根据需求创建逻辑卷（重点）
- 能够根据需求动态扩容逻辑卷（重点）
- 熟练使用逻辑卷相关命令（pvcreate/vgcreate/lvcreate等）

理论储备

一、硬盘的基本知识

1.了解硬盘的接口类型

IDE ——> SATA I/II/III 个人pc机 SCSI ——> SAS 服务器上

2. 硬盘命名方式

OS	IDE(并口)	SATA(串口)	SCSI
RHEL5	/dev/hda	/dev/sda	/dev/sda
RHEL6	/dev/sda	/dev/sda	/dev/sda
RHEL7	/dev/sda	/dev/sda	/dev/sda

3. 磁盘设备的命名

/dev/sda2

s=硬件接口类型（sata/scsi）,d=disk（硬盘）,a=第1块硬盘（b，第二块），2=第几个分区 /dev/hd h=IDE硬盘 /dev/hdd3 /dev/vd v=虚拟硬盘 /dev/vdf7

4. HP服务器硬盘

/dev/cciss/c0d0 /dev/cciss/c0d0p1 c0第一个控制器, d0第一块磁盘, p1分区1 /dev/cciss/c0d0p2 c0第一个控制器, d0第一块磁盘, p2分区2

5. 硬盘的分区方式

MBR <2TB fdisk 4个主分区或者3个主分区+1个扩展分区（N个逻辑分区）

MBR(Master Boot Record)的缩写，由三部分组成，即：

1. Bootloader（主引导程序）=446字节 硬盘第一个扇区=512字节
    - 引导操作系统的主程序
2. DPT分区表（Disk Partition Table）=64字节
    - 分区表保存了硬盘的分区信息，操作系统通过读取分区表内的信息，就能够获得该硬盘的分区信息
    - 每个分区需要占用16个字节大小，保存有文件系统标识、起止柱面号、磁头号、扇区号、起始扇区位置（4个字节）、分区总扇区数目（4个字节）等内容
    - 分区表中保存的分区信息都是主分区与扩展分区的分区信息，扩展分区不能直接使用，需要在扩展分区内划分一个或多个逻辑分区后才能使用
    - 逻辑分区的分区信息保存在扩展分区内而不是保存在MBR分区表内，这样，就可以突破MBR分区表只能保存4个分区的限制
3. 硬盘有效标志（校验位）=2个字节

GPT >2TB gdisk(parted) 128个主分区

注意：从MBR转到GPT，或从GPT转换到MBR会导致**数据全部丢失**!

扩展阅读：http://www.eassos.cn/jiao-cheng/ying-pan/mbr-vs-gpt.php

二、硬盘的工作原理

了解

三、基本分区管理

1. 磁盘划分思路

- 进入分区表 新建分区 fdisk /dev/sdb
- 更新分区表<刷新分区表>
- 格式化分区——>文件系统 mkfs.ext4 /dev/sdb1
- 挂载使用——>mount【开机自动挂载|autofs自动挂载】

2. fdisk分区

2.1 使用fdisk分区

```
# lsblk
# df -h  查看正在挂载的设备情况
# fdisk -l      查看当前系统的所有设备分区情况
# fdisk  /dev/sdb

硬盘容量  =  柱面数  ×  盘面数（磁头数）  ×  扇区数  ×  扇区大小（一般为512字节）
Disk /dev/sda: 26.8 GB, 26843545600 bytes   磁盘空间
 255 heads, 63 sectors/track, 3263 cylinders
 255磁头    63个扇区 每 磁道   3263个圆柱体
Units = cylinders of 16065 * 512 = 8225280 bytes       单元
Sector size (logical/physical): 512 bytes / 512 bytes
扇区大小（逻辑/物理）  都是512字节。
I/O size (minimum/optimal): 512 bytes / 512 bytes
I/O 大小（最小/最大）  都是512字节。
Disk identifier: 0x00030124   设备识别码

 启动设备加*          起始         结束      块           id     系统
 Device Boot      Start         End      Blocks     Id  System
/dev/sda1   *           1        2497    20051968   83  Linux
/dev/sda2            2497        2611      916481    5  Extended  扩展分区
/dev/sda3            2612        3263     5237190   83  Linux
/dev/sda5            2497        2611      916480   82  Linux swap / Solaris

Command(m for help): m                          输出帮助信息
Commandaction
   a  toggle a bootable flag                    设置启动分区
   b  edit bsd disklabel                        编辑分区标签
   c  toggle the dos compatibility flag
   d  delete a partition                        删除一个分区
   l  list known partition types                列出分区类型
   m  print this menu                           帮助
   n  add a new partition                       建立一个新的分区
   o  create a new empty DOS partition table    创建一个新的空白DOS分区表
   p  print the partition table                 打印分区表
   q  quit without saving changes               退出不保存设置
   s   createa new empty Sun disklabel          创建一个新的空的SUN标示
   t   changea partition's system id            改变分区的类型
   u   changedisplay/entry units                改变显示的单位
   v   verifythe partition table                检查验证分区表
   w  write table to disk and exit              保存分区表
```

总结：

1. 最多只能分4个主分区，主分区编号1-4

2. 逻辑分区大小总和不能超过扩展分区大小，逻辑分区分区编号从5开始
3. 如果删除扩展分区，下面的逻辑卷分区也被删除
4. 扩展分区的分区编号（1-4）

任务1：

添加一块硬盘，需要将其分两个分区，分别格式化成ext4和vfat格式文件系统使用，最终需要使用2G空间。

```
思路：
1. 增加一块硬盘
2. 使用fdisk命令进行分区
3. 格式化指定分区
4. 创建一个空的目录作为挂载点
5. 挂载使用

步骤：
1. 增加硬盘
增加完硬盘记得重启系统
# lsblk  查看硬盘是否添加成功
...
sdb                       8:16   0   10G  0 disk
[root@web ~]# fdisk -l /dev/sdb
Disk /dev/sdb: 10.7 GB, 10737418240 bytes
255 heads, 63 sectors/track, 1305 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000
2. 使用fdisk命令分区
[root@web ~]# fdisk /dev/sdb
Command (m for help): p       打印分区表信息

Disk /dev/sdb: 10.7 GB, 10737418240 bytes
255 heads, 63 sectors/track, 1305 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x31dd29ec

   Device Boot      Start         End      Blocks   Id  System

Command (m for help): n       创建新分区
Command action
   e   extended      扩展分区
   p   primary partition (1-4)   主分区
p
Partition number (1-4): 1     选择主分区编号
First cylinder (1-1305, default 1):    起始柱面默认即可（直接回车）
Using default value 1
Last cylinder, +cylinders or +size{K,M,G} (1-1305, default 1305): +1G  分区大小1G

Command (m for help): p
```

```
Disk /dev/sdb: 10.7 GB, 10737418240 bytes
255 heads, 63 sectors/track, 1305 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x31dd29ec

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1               1         132     1060258+  83  Linux

Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 2
First cylinder (133-1305, default 133):
Using default value 133
Last cylinder, +cylinders or +size{K,M,G} (133-1305, default 1305): +1G

Command (m for help): p

Disk /dev/sdb: 10.7 GB, 10737418240 bytes
255 heads, 63 sectors/track, 1305 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x31dd29ec

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1               1         132     1060258+  83  Linux
/dev/sdb2             133         264     1060290   83  Linux

Command (m for help): w       保存退出
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
```

3. 再次查看分区情况

```
# lsblk
sdb                     8:16   0   10G  0 disk
├─sdb1                  8:17   0    1G  0 part
└─sdb2                  8:18   0    1G  0 part
```

4. 刷新分区表信息

```
[root@web ~]# partx -a /dev/sdb
BLKPG: Device or resource busy
error adding partition 1
BLKPG: Device or resource busy
error adding partition 2
```

5. 格式化分区

```
[root@web ~]# mkfs.ext4 /dev/sdb1
[root@web ~]# yum -y install dosfstools   需要mkfs.vfat命令，安装软件
[root@web ~]# mkfs.vfat /dev/sdb2

6. 创建新的挂载点
[root@web ~]# mkdir /u01
[root@web ~]# mkdir /u02

7. 挂载使用
[root@web ~]# mount /dev/sdb1 /u01
[root@web ~]# mount /dev/sdb2 /u02
```

总结：

1. 扩展分区的大小决定了所有逻辑分区的大小
2. 删除扩展分区后下面的逻辑分区都被删除
3. 分完区后需要手动刷新分区表，如果刷新不成功需要重启操作系统
4. 创建分区的时候尽可能注意分区序号的连续性

2.2 挂载分区设备

手动挂载：

```
mount   [options]      需要挂载的设备       挂载点
特点：系统重启后需要重新挂载；手动卸载后需要手动挂载

-o:挂载选项  ro,sync,rw,remount
-t:文件系统类型
mount -t nfs=mount.nfs
mount -t cifs=mount.cifs

10.1.1.2  /share   [smb]

mount.cifs -o user=user01,password=123 //10.1.1.2/smb /u01
mount.nfs 10.1.1.2:/share /u02

[root@MissHou ~]# mount -o remount,ro /u02       //可以是挂载点也可以是设备
remount:重新挂载一个正在挂载的设备

# mount -o remount,ro /dev/sdb1
# mount -o remount,ro /u01
注意：后面可以根挂载点也可以跟设备本身


挂载设备：真实设备、设备UUID，设备的卷标
/dev/sdb
/dev/sdb1
[root@MissHou ~]# blkid /dev/sdb1             //查看设备的UUID和文件系统类型
/dev/sdb1: UUID="FD3A-F14D" TYPE="vfat"
[root@MissHou ~]# blkid /dev/sdb2
/dev/sdb2: UUID="f1cc2198-7e5f-4408-9c74-9b93d4716d8d" TYPE="ext4"
[root@server ~]# e2label /dev/sdb1 DISK1
```

```
说明：e2label只能够对ext2~ext4的文件系统设置卷标

[root@MissHou ~]# e2label /dev/sdb2 disk2
[root@MissHou ~]# blkid /dev/sdb2
/dev/sdb2: UUID="f1cc2198-7e5f-4408-9c74-9b93d4716d8d" TYPE="ext4" LABEL="disk2"

卸载设备：umount
[root@MissHou ~]# umount /u01
[root@MissHou ~]# umount /dev/sdb2
```

开机自动挂载：

操作系统启动流程：

1. 硬件初始化 硬盘、内存、。。。HD
2. 系统初始化 /sbin/init—>xxxxxx/etc/fstab

```
# vim /etc/fstab         //开机自动挂载
UUID="9bf6b9f7-92ad-441b-848e-0257cbb883d1" /mnt/disk1   auto    defaults    0 0
UUID="4d26172c-7aff-4388-baa5-c6756c014d52" /mnt/disk2    ext4    ro    0 0
# mount -a
特点：系统重启后自动挂载；手动卸载后重启会自动挂载或者使用mount -a自动挂载

/etc/fstab文件：
格式：
要挂载的资源路径 挂载点    文件系统类型    挂载选项   dump备份支持   文件系统检测
UUID=289370eb-9459-42a8-8cee-7006507f1477   /       ext4    defaults        1 1
1段：挂载的设备（磁盘设备的文件名或设备的卷标或者是设备的UUID）
2段：挂载点（建议用一个空目录），建议不要将多个设备挂载到同一个挂载点上
3段：文件系统类型（ext3、ext4、vfat、ntfs（安装软件包）、swap等等）
4段：挂载选项
async/sync  异步/同步：
auto/noauto     自动/非自动：
rw/ro   读写/只读：
exec/noexec     可被执行/不可被执行：
remount     重新挂在一个已经挂载的文件系统，常用于修改挂载参数
user/nouser     允许/不允许其他普通用户挂载：
suid/nosuid     具有/不具有suid权限：该文件系统是否允许SUID的存在。
usrquota    这个是在启动文件系统的时候，让其支持磁盘配额，这个是针对用户的。
grpquota    支持用户组的磁盘配额。
....
defaults 同时具有rw, dev, exec, acl, async,nouser等参数。

mount -a  重新读取/etc/fstab文件内容

man mount 可以找到详细信息

5段：是否支持dump备份。//dump是一个用来备份的命令，0代表不要做dump备份，1代表要每天进行dump的动作，2也代表其他不定日期的dump备份。通常这个数值不是0就是1。数字越小优先级越高。
```

6段：是否用 fsck 检验扇区。//开机的过程中，系统默认会用fsck检验文件系统是否完整。0是不要检验，1表示最先检验(一般只有根目录会设定为1)，2也是要检验，只是1是最先，2是其次才进行检验。

```
# fsck -f /dev/sdb2      强制检验/dev/sdb2上文件系统
```

说明：
要挂载的资源路径可以是文件系统的UUID，设备路径，文件系统的标签 ，光盘镜像文件（iso），亦或是来自网络的共享资源等

建议：
/etc/rc.local 操作系统启动后读取的最后一个文件

```
vim /etc/rc.local
...
/bin/mount -o noexec,ro /dev/sdb1 /u01
```

自动挂载 Automount：

特点：挂载是由访问产生；卸载是由超时产生；依赖于后台的autofs服务
思路：
1. 所有的监控都是由一个程序完成 autofs
2. 服务启动后才会监控挂载的设备
3. 修改配置文件来指定需要监控的设备

需求1：让系统自动挂载/dev/sdb1设备，如果2分钟没有被用自动卸载

步骤：
1）安装autofs软件

```
[root@server ~]# rpm -q autofs
package autofs is not installed
[root@server ~]#
[root@server ~]# yum list|grep autofs
autofs.x86_64                           1:5.0.5-88.el6                    local
libsss_autofs.x86_64                    1.9.2-129.el6                     local
[root@server ~]# yum -y install autofs

[root@server ~]# rpm -q autofs
autofs-5.0.5-88.el6.x86_64
```

2）修改配置文件（指定需要监控的设备和挂载的目录）

```
vim /etc/auto.master    //定义一级挂载点/u01和子配置文件
/u01    /etc/auto.test  -t 120 或者 --timeout 120  单位秒    （设置超时时间去卸载）

vim /etc/auto.test         //子配置文件自己创建，定义二级挂载点和需要挂载的设备
test  -fstype=ext4,ro   :/dev/sdb5
```

3）重启服务

```
[root@MissHou ~]# service autofs restart
Stopping automount:                                        [  OK  ]
Starting automount:                                        [  OK  ]
```

```
[root@MissHou ~]#

4）测试验证
[root@server ~]# ls /u01/test
[root@server ~]# df -h
Filesystem                    Size  Used Avail Use% Mounted on
/dev/mapper/VolGroup-lv_root   18G  6.6G  9.9G  41% /
tmpfs                         491M     0  491M   0% /dev/shm
/dev/sda1                     485M   33M  427M   8% /boot
/dev/sr0                      4.2G  4.2G     0 100% /mnt


后续补充:
如果想要将/dev/sdb5挂载到/u01下，怎么做?
vim /etc/auto.master
/-    /etc/auto.test

vim /etc/auto.test
/u01  -fstype=ext4 :/dev/sdb5

需求2:将10.1.1.1(node1)上的共享目录/share/nfs挂载到10.1.1.2(server)本地的/notes下面

环境:
node1: 10.1.1.1 搭建NFS服务，共享/share/nfs/xxx
server:10.1.1.2   使用autofs自动挂载server上共享目录
步骤:
1. node1端共享文件
1)修改/etc/exports文件
[root@node1 ~]# mkdir /share/nfs -p
[root@node1 ~]# touch /share/nfs/file{1..5}
[root@node1 ~]# vim /etc/exports
[root@node1 ~]# cat /etc/exports
/share/nfs  10.1.1.2/24(rw)
2）重启nfs服务
[root@node1 ~]# service rpcbind restart
[root@node1 ~]# service nfs restart

2. server端使用autofs方式自动挂载
[root@server ~]# showmount -e 10.1.1.1
Export list for 10.1.1.1:
/share/nfs 10.1.1.2/24
[root@server ~]#
[root@server ~]# vim /etc/auto.master
/- /etc/auto.notes  --timeout 60
[root@server ~]# vim /etc/auto.notes
/notes   -nfs,ro  10.1.1.1:/share/nfs

3）重启服务
[root@server ~]# service autofs restart
Stopping automount:                                        [  OK  ]
Starting automount:                                        [  OK  ]

4）测试验证
```

```
[root@server ~]# ls /notes
file1  file2  file3  file4  file5
[root@server ~]# df -h
Filesystem                    Size  Used Avail Use% Mounted on
/dev/mapper/VolGroup-lv_root   18G  6.6G  9.9G  41% /
tmpfs                         491M     0  491M   0% /dev/shm
/dev/sda1                     485M   33M  427M   8% /boot
/dev/sr0                      4.2G  4.2G     0 100% /mnt
10.1.1.1:/share/nfs            18G  4.1G   13G  25% /notes

注意：一级和二级挂载点可以不用创建，autofs可以让我们临时使用
```

总结：挂载设备的三种方式

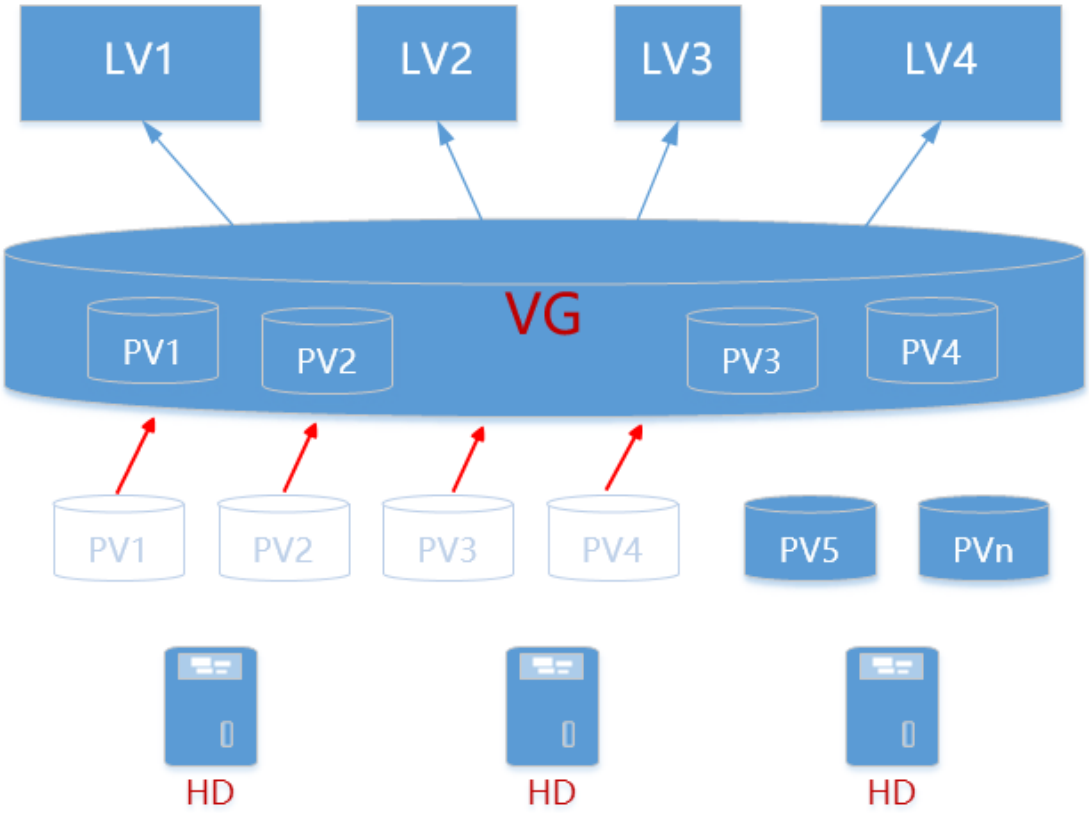
1. 设备要被挂载，必须要有文件系统类型(mkfs.类型)
2. 手动挂载mount mount -o 挂载选项(ro,rw,auto,noexec...) 需要挂载设备 挂载点
    - 重启需要再次重新挂载
3. 开机自动挂载
    - /etc/fstab 修改特别小心（不建议）
    - 修改/etc/rc.local文件 启动后最后读取文件，执行该文件必须要有可执行权限
4. autofs自动挂载
    - 使用时触发自动挂载
    - 超时自动卸载
    - 如何配置设备的autofs自动挂载
        1）安装autofs软件
        2）修改/etc/auto.master文件 定义1级挂载点和超时时间，子配置文件
        3）创建刚刚定义的子配置文件 二级挂载点和需要挂载设备以及文件系统类型及挂载方式
        4）启动服务，测试验证

四、逻辑卷管理（重点）

1. 逻辑卷介绍

逻辑卷： 逻辑卷（LVM）：它是Linux环境下对磁盘分区进行管理的一种机制，它是建立在**物理存储设备**之上的一个抽象层，优点在于**灵活**管理。 **特点：** 1、**动态在线扩容(重点)** 2、离线裁剪 3、数据条带化 4、数据镜像

2. 逻辑卷基本概念



- 物理卷（Physical Volume，PV）

物理卷是底层真正提供容量，存放数据的设备,它可以是整个硬盘、硬盘上的分区等。

- 卷组（Volume Group, VG）

卷组建立在物理卷之上，它由一个或多个物理卷组成。即把物理卷整合起来提供容量分配。 一个LVM系统中可以只有一个卷组，也可以包含多个卷组。

- 逻辑卷（Logical Volume, LV）

逻辑卷建立在卷组之上，它是从卷组中“切出”的一块空间。它是最终用户使用的逻辑设备。逻辑卷创建之后，其大小可以伸缩。

- 物理区域 PE（physical extent）

每一个物理卷被划分为称为PE(Physical Extents)的基本单元，具有唯一编号的PE是能被LVM寻址的最小单元。PE的大小可指定，默认为4 MB。 PE的大小一旦确定将不能改变，同一个卷组中的所有物理卷的PE的大小是一致的。
4MB=4096kb=4096kb/4kb=1024个block

说明：

1. 硬盘读取数据最小单位1个扇区512字节
2. 操作读取数据最小单位1个数据块=8*512字节=4096字节=4KB
3. lvm寻址最小单位1个PE=4MB

- 逻辑区域 LE（logical extent）

逻辑卷也被划分为被称为LE(Logical Extents) 的可被寻址的基本单位。在同一个卷组中，LE的大小和PE是相同的，并且一一对应。

真实的物理设备——>逻辑上(命令创建)——>物理卷（pv）——>卷组（vg）——>逻辑卷（lv）——>逻辑卷格式化——>挂载使用

3. 逻辑卷LVM应用

3.1 逻辑卷创建

需求：创建一个2.5G大小的逻辑卷

```
思路：
1. 物理的设备
2. 将物理设备做成物理卷
3. 创建卷组并将物理卷加入其中
4. 创建逻辑卷
5. 格式化逻辑卷
6. 挂载使用

步骤：
1. 物理设备
sdb                           8:16   0   10G  0 disk
├─sdb1                        8:17   0    2G  0 part
├─sdb2                        8:18   0    2G  0 part
├─sdb3                        8:19   0    1K  0 part
├─sdb5                        8:21   0    2G  0 part
├─sdb6                        8:22   0    2G  0 part
└─sdb7                        8:23   0    2G  0 part

2. 创建物理卷
[root@server ~]# pvcreate /dev/sdb1 /dev/sdb2
查看物理卷：
[root@server ~]# pvs    简单查看
  PV         VG        Fmt  Attr PSize  PFree
  /dev/sdb1            lvm2 a--   2.01g 2.01g
  /dev/sdb2            lvm2 a--   2.01g 2.01g
[root@server ~]# pvdisplay /dev/sdb1      详细查看

3. 创建卷组并将物理卷加入其中
[root@server ~]# vgcreate vg01 /dev/sdb1 /dev/sdb2
  Volume group "vg01" successfully created
查看卷组信息：
[root@server ~]# vgs vg01     简单查看
  VG   #PV #LV #SN Attr   VSize VFree
  vg01   2   0   0 wz--n- 4.01g 4.01g

[root@server ~]# vgdisplay vg01  详细查看

4. 创建逻辑卷
[root@server ~]# lvcreate -n lv01 -L 2.5G vg01
  Logical volume "lv01" created
在操作系统层面映射两个地方：
[root@server ~]# ll /dev/mapper/vg01-lv01
```

```
lrwxrwxrwx 1 root root 7 Jan  7 11:16 /dev/mapper/vg01-lv01 -> ../dm-2
[root@server ~]# ll /dev/vg01/lv01
lrwxrwxrwx 1 root root 7 Jan  7 11:16 /dev/vg01/lv01 -> ../dm-2
查看逻辑卷的信息:
[root@server ~]# lvdisplay /dev/vg01/lv01

-n: 指定逻辑卷的名字
-L: 指定逻辑卷的大小
-l: 指定逻辑卷的大小
举例:
-l 100          100个PE，每个PE大小默认4M，故逻辑卷大小为400M
-l 50%free      卷组剩余空间的50%
[root@server ~]# vgs vg01
  VG   #PV #LV #SN Attr   VSize VFree
  vg01   1   1   0 wz--n- 2.00g 516.00m

创建大小为200M的逻辑卷lv02;每个PE为4M，-l50指定50个PE,大小为200M
[root@server ~]# lvcreate -n lv02 -l50 vg01
  Logical volume "lv02" created
[root@server ~]# vgs vg01
  VG   #PV #LV #SN Attr   VSize VFree
  vg01   1   2   0 wz--n- 2.00g 316.00m
[root@server ~]# lvs /dev/vg01/lv02
  LV   VG   Attr       LSize   Pool Origin Data%  Move Log Cpy%Sync Convert
  lv02 vg01 -wi-a----- 200.00m

创建大小为剩余卷组vg01空间的50%的逻辑卷lv03
[root@server ~]# lvcreate -n lv03 -l50%free vg01
  Logical volume "lv03" created
[root@server ~]# lvs /dev/vg01/lv03
  LV   VG   Attr       LSize   Pool Origin Data%  Move Log Cpy%Sync Convert
  lv03 vg01 -wi-a----- 156.00m
[root@server ~]# vgs vg01
  VG   #PV #LV #SN Attr   VSize VFree
  vg01   1   3   0 wz--n- 2.00g 160.00m

5. 格式化逻辑卷
[root@server ~]# mkfs.ext4 /dev/vg01/lv01
6. 挂载使用
1）创建一个空的挂载点
2）挂载使用
[root@server ~]# mount /dev/vg01/lv01 /u01
```

3.2 逻辑卷动态扩容

需求：将/u01目录动态扩容到3G

```
思路:
1. 查看/u01目录所对应的逻辑卷是哪一个  /dev/mapper/vg02-lv01
2. 查看当前逻辑卷所在的卷组vg01剩余空间是否足够
3. 如果vg01空间不够，得先扩容卷组，再扩容逻辑卷
4. 如果vg01空间足够，直接扩容逻辑卷
```

步骤：
1. 查看/u01目录属于哪个卷组
```
[root@web ~]# df -h
Filesystem                Size  Used Avail Use% Mounted on
/dev/mapper/vg01-lv_root   18G  5.9G   11G  36% /
tmpfs                     931M     0  931M   0% /dev/shm
/dev/sda1                 291M   33M  244M  12% /boot
/dev/sr0                  4.2G  4.2G     0 100% /mnt
/dev/mapper/vg02-lv01     1.5G   35M  1.4G   3% /u01

[root@web ~]# lvs
  LV      VG   Attr       LSize  Pool Origin Data%  Move Log Cpy%Sync Conver

  lv01    vg02 -wi-ao----  1.50g
```
2. 卷组的剩余空间
```
[root@web ~]# vgs
  VG   #PV #LV #SN Attr   VSize  VFree
  vg01   1   2   0 wz--n- 19.70g      0
  vg02   1   1   0 wz--n-  2.00g 516.00m
```
结果：当前卷组空间不足我扩容

3. 扩容逻辑卷所在的卷组
1）首先得有物理设备 /dev/sdb5
2）将物理设备做成物理卷
```
[root@web ~]# pvcreate /dev/sdb5
  Physical volume "/dev/sdb5" successfully created
[root@web ~]# pvs
  PV         VG   Fmt  Attr PSize  PFree
  /dev/sda2  vg01 lvm2 a--  19.70g      0
  /dev/sdb5       lvm2 a--   2.01g   2.01g
  /dev/sdb6  vg02 lvm2 a--   2.00g 516.00m
```
3）将物理卷加入到卷组中（卷组扩容）
```
[root@web ~]# vgextend vg02 /dev/sdb5
  Volume group "vg02" successfully extended
[root@web ~]# pvs
  PV         VG   Fmt  Attr PSize  PFree
  /dev/sda2  vg01 lvm2 a--  19.70g      0
  /dev/sdb5  vg02 lvm2 a--   2.00g   2.00g
  /dev/sdb6  vg02 lvm2 a--   2.00g 516.00m
```

注意：
正常情况下，应该先将/dev/sdb5物理设备创建为物理卷再加入到卷组中；如果直接加入卷组，系统会自动帮你将其做成物理卷。

```
[root@web ~]# vgs
  VG   #PV #LV #SN Attr   VSize  VFree
  vg01   1   2   0 wz--n- 19.70g    0
  vg02   2   1   0 wz--n-  4.01g 2.51g
```

4. 扩容逻辑卷
```
[root@web ~]# lvextend -L 3G /dev/vg02/lv01     -L 3G最终的大小
```
或者
```
[root@web ~]# lvextend -L +1.5G /dev/vg02/lv01   -L +1.5G 扩容1.5G
```

```
5. 查看结果
[root@web ~]# lvs
  LV      VG   Attr       LSize  Pool Origin Data%  Move Log Cpy%Sync Convert
  lv_root vg01 -wi-ao---- 17.70g
  lv_swap vg01 -wi-ao----  2.00g
  lv01    vg02 -wi-ao----  3.00g    已经扩容到了3G

[root@web ~]# df -h
Filesystem                Size  Used Avail Use% Mounted on
/dev/mapper/vg01-lv_root   18G  5.9G   11G  36% /
tmpfs                     931M     0  931M   0% /dev/shm
/dev/sda1                 291M   33M  244M  12% /boot
/dev/sr0                  4.2G  4.2G     0 100% /mnt
/dev/mapper/vg02-lv01     1.5G   35M  1.4G   3% /u01     实际并没有改变

6. 同步文件系统
[root@web ~]# resize2fs /dev/vg02/lv01
7. 再次查看验证
[root@web ~]# df -h
Filesystem                Size  Used Avail Use% Mounted on
/dev/mapper/vg01-lv_root   18G  5.9G   11G  36% /
tmpfs                     931M     0  931M   0% /dev/shm
/dev/sda1                 291M   33M  244M  12% /boot
/dev/sr0                  4.2G  4.2G     0 100% /mnt
/dev/mapper/vg02-lv01     3.0G   35M  2.8G   2% /u01     扩容成功
```

离线裁剪逻辑卷（了解）

```
[root@server ~]# umount /data/
[root@server ~]# e2fsck -f /dev/vg01/lv01    检验文件系统
[root@server ~]# resize2fs /dev/vg01/lv01 2G 裁剪文件系统到2G
[root@server ~]# lvreduce /dev/vg01/lv01 -L 2G  裁剪逻辑卷
[root@server ~]# mount /dev/vg01/lv01 /data     挂载使用
```

3.3 逻辑卷相关命令

```
创建物理卷:pvcreate
pvcreate /dev/sdb1
创建卷组:vgcreate
vgcreate vg01 /de/sdb1
创建逻辑卷:lvcreate
lvcreate -n lv01 -L 1G vg01
lvcreate -n lv01 -l 100 vg01
lvcreate -n lv01 -l 100%free vg01

删除逻辑卷:lvremove
lvremove /dev/vg01/lv01
删除卷组:vgremove
vgremove vg01
```

```
说明：卷组里的物理卷没有被使用才可以直接删除卷组
删除物理卷:pvremove
pvremove /dev/sdb1

扩容卷组:vgextend
vgextend vg01 /dev/sdb2
扩容逻辑卷:lvextend
lvextend /dev/vg01/lv01 -L +2G
同步文件系统:
resize2fs /dev/vg01/lv01

裁剪卷组:vgreduce
vgreduce vg01 /dev/sdb2
裁剪逻辑卷:lvreduce
```

实战演练

```
思路:
0. 准备好物理设备，并创建一个逻辑卷，大小根据mysql数据库的实际大小再大一些，挂载到系统中
1. 最好在系统维护时间操作（23:00-8:00）
2. 先停止前端应用 LAMP apache
3. 停止mysql数据库(建议备份mysql数据库)
4. 迁移mysql数据文件

步骤:
1. 添加一块物理硬盘[是否需要重启开服务器是否支持热插拔]
sdb                             8:16   0   10G  0 disk
├─sdb1                          8:17   0    2G  0 part
├─sdb2                          8:18   0    2G  0 part
├─sdb3                          8:19   0    1K  0 part
├─sdb5                          8:21   0    2G  0 part
├─sdb6                          8:22   0    2G  0 part
└─sdb7                          8:23   0    2G  0 part
2. 创建大小为8G的逻辑卷
1）创建物理卷
[root@server ~]# pvcreate /dev/sdb[12567]
2）创建卷组vg_mysql
[root@server ~]# vgcreate vg_mysql /dev/sdb[12567]
[root@server ~]# vgs vg_mysql
  VG       #PV #LV #SN Attr   VSize VFree
  vg_mysql   5   0   0 wz--n- 9.98g 9.98g
3）创建逻辑卷lv_mysql
[root@server ~]# lvcreate -n lv_mysql -L 8G vg_mysql
[root@server ~]# lvs /dev/vg_mysql/lv_mysql
  lv_mysql vg_mysql -wi-a----- 8.00g

4）格式化为ext4文件系统
[root@server ~]# mkfs.ext4 /dev/vg_mysql/lv_mysql

5）挂载使用
a. 创建一个空的挂载点/u01
[root@server ~]# mkdir /u01
```

```
b. 挂载逻辑卷lv_mysql到/u01目录
[root@server ~]# mount /dev/vg_mysql/lv_mysql /u01


3. 停止前端web服务
[root@server ~]# service apache stop

4. 停止mysql数据库
[root@server ~]# service mysql stop

5. 备份mysql数据库到另外一台备份机（实验环境不是必须）
备份机：10.1.1.2   备份目录:/backup
[root@server ~]# rsync -av /usr/local/mysql 10.1.1.2:/backup/

6. 将/usr/local/mysql/目录里的所有数据文件同步到逻辑卷上，即/u01目录
[root@server ~]# rsync -av /usr/local/mysql/ /u01
查看是否同步完成：
[root@server ~]# ls /u01
[root@server ~]# du -sh /u01     查看大小是否和原来mysql数据库大小一致

7. 卸载逻辑卷
[root@server ~]# umount /u01

8. 删除/usr/local/mysql/目录里原来的数据文件
注意：删之前一定要确定成功备份了！！！
[root@server ~]# rm -rf /usr/local/mysql/*

9. 挂载逻辑卷lv_mysql到mysql的安装目录/usr/local/mysql
[root@server ~]# mount /dev/vg_mysql/lv_mysql /usr/local/mysql/
开机自动挂载：
vim /etc/rc.local
...
mount /dev/vg_mysql/lv_mysql /usr/local/mysql/

10. 启动数据库
[root@server ~]# service mysql start

11. 启动web服务
[root@server ~]# service apache start

12.测试验证
访问之前的网站看是否可以正常访问
```

课后扩展补充

一、 扩容swap空间

方法1：增加一个硬盘或者分区来扩容swap空间

```
查看swap空间大小：
[root@web ~]# free -m
             total       used       free     shared    buffers     cached
Mem:          1861        646       1215          0          9         60
```

```
-/+ buffers/cache:        576       1285
Swap:         2047          0       2047
[root@web ~]# swapon -s
Filename                        Type            Size    Used    Priority
/dev/dm-1                                       partition       2097144 0       -1

[root@web ~]# mkswap /dev/sdb7
Setting up swapspace version 1, size = 2104476 KiB
no label, UUID=485ff8ad-a636-4556-a2e7-4ee9efc78afb
[root@web ~]# blkid /dev/sdb7
/dev/sdb7: UUID="485ff8ad-a636-4556-a2e7-4ee9efc78afb" TYPE="swap"

//激活swap分区。swap空间不能手动挂载
[root@server ~]# swapon /dev/sdb7

[root@server ~]# swapon -s
Filename                        Type            Size    Used    Priority
/dev/dm-1                                       partition       2031608 0       -1
/dev/sdb7                                       partition       2064312 0       -2
```

方法2：使用dd命令模拟大文件来扩容swap

```
[root@server ~]# dd if=/dev/sr0 of=/rhel6.iso
[root@server ~]# dd if=/dev/sda1 of=/tmp/bak.boot
[root@server ~]#
[root@server ~]# dd if=/dev/zero of=/dev/sda1 bs=1M count=100  //不要执行

if=源文件
of=目标文件
bs=复制数据的大小
count=复制的个数

注意：
1. 一般可以使用dd命令做块设备文件的备份
2. /dev/zero 特殊设备，一般用来模拟一个大文件，源源不断的二进制的bit流；
   /dev/null  空设备，类似黑洞

步骤：
1. 使用dd命令模拟大文件
# dd if=/dev/zero of=/tmp/swapfile bs=1M count=1024
2. 格式化大文件
[root@server ~]# mkswap /tmp/swapfile

3.激活大文件
[root@server ~]# swapon -p 1 /tmp/swapfile
-p：指定优先级，数字越大优先级越高，0-32767
4. 查看
[root@server ~]# swapon -s
Filename                        Type            Size    Used    Priority
/tmp/swapfile                                   file    1048568 0       1

如果开机自动挂载，需要修改文件：/etc/fstab
```

```
/swap_file  swap swap defautls,pri=1 0 0
```

二、逻辑卷实现条带化

条带化：

把保存到逻辑卷的数据分成n等分，分别写到不同的物理卷，可以提高数据的读写效率；

如果任何一个涉及到的物理卷出现故障，数据都会无法恢复。

```
sdc                           8:32   0   20G  0 disk
├─sdc1                        8:33   0    2G  0 part
└─sdc2                        8:34   0    2G  0 part
```

创建物理卷

```
[root@server ~]# pvcreate /dev/sdc[12]
```

查看物理卷

```
[root@server ~]# pvs
/dev/sdc1            lvm2 a--   2.01g  2.01g
/dev/sdc2            lvm2 a--   2.01g  2.01g
```

创建卷组：

```
[root@server ~]# vgcreate vg01 /dev/sdc[12]

[root@web ~]# pvs /dev/sdc[12]
  PV         VG   Fmt  Attr PSize PFree
 /dev/sdc1  vg01      lvm2 a--   2.00g  2.00g
 /dev/sdc2  vg01      lvm2 a--   2.00g  2.00g
```

创建实现条带化的逻辑卷：

```
[root@server ~]# lvcreate -n lv01 -L 1G vg01 -i 2 /dev/sdc[12]
  Using default stripesize 64.00 KiB
  Logical volume "lv01" created

[root@server ~]# lvs /dev/vg01/lv01
  LV   VG   Attr       LSize Pool Origin Data%  Move Log Cpy%Sync Convert
  lv01 vg01 -wi-a----- 1.00g
[root@server ~]# pvs /dev/sdc[12]
  PV         VG   Fmt  Attr PSize PFree
  /dev/sdc1  vg01 lvm2 a--  2.00g 1.50g
  /dev/sdc2  vg01 lvm2 a--  2.00g 1.50g
```

-i 参数：给出条带化的数量

格式化挂载使用：

```
[root@server ~]# mkfs.ext4 /dev/vg01/lv01
[root@server ~]# mount /dev/vg01/lv01 /u01
```

测试：

```
[root@server ~]# yum -y install sysstat
[root@server ~]# iostat -m -d /dev/sdc[12] 2
-d 查看磁盘
-m 以什么速度显示，每秒M
 2 每隔2s显示一次
   如果后面还有数字则代表总共显示多少次

[root@server ~]# dd if=/dev/zero of=/u01/test bs=1M count=1000    模拟写数据
[root@server ~]# iostat -m -d /dev/sdc[12] 1
。。。
Device:            tps    MB_read/s    MB_wrtn/s    MB_read    MB_wrtn
sdb5            178.00         0.00        52.00          0         52
sdb6            177.00         0.00        52.00          0         52
```

三、逻辑卷实现镜像

逻辑卷实现镜像：
镜像：对某个逻辑卷的数据做镜像，起到数据备份的作用。

当前环境：

```
├─sdb7                    8:23   0    2G  0 part
└─sdb8                    8:24   0    2G  0 part
```

创建物理卷：

```
[root@web ~]# pvcreate /dev/sdb[78]
[root@web ~]# pvs /dev/sdb[78]
  PV         VG   Fmt  Attr PSize PFree
  /dev/sdb7       lvm2 a--  2.01g 2.01g
  /dev/sdb8       lvm2 a--  2.01g 2.01g
```

将物理卷加入到vg02卷组：

```
[root@web ~]# vgextend vg02 /dev/sdb[78]
  Volume group "vg02" successfully extended
[root@web ~]# vgs vg02
  VG   #PV #LV #SN Attr   VSize VFree
  vg02   4   1   0 wz--n- 8.02g 6.02g
```

创建实现镜像的逻辑卷：

```
[root@web ~]# lvcreate -n lv02 -L 2G vg02 -m 1 /dev/sdb[78]
  Logical volume "lv02" created
```

-m参数：给出镜像的个数；1表示1个镜像

```
[root@web ~]# lvs
  LV       VG   Attr       LSize  Pool Origin Data%  Move Log       Cpy%Sync Convert
  lv_root  vg01 -wi-ao---- 17.70g
  lv_swap  vg01 -wi-ao----  2.00g
  lv01     vg02 -wi-ao----  2.00g
```

```
  lv02     vg02 mwi-a-m---  2.00g                          lv02_mlog     38.67
  lv-mysql vg03 -wi-ao---- 10.00g
```

说明： Cpy%Sync 53.52该值是100%说明复制ok

创建后：

```
[root@web ~]# pvs /dev/sdb[78]
  PV         VG   Fmt  Attr PSize PFree
  /dev/sdb7  vg02 lvm2 a--  2.00g 4.00m
  /dev/sdb8  vg02 lvm2 a--  2.00g    0
```

格式化逻辑卷：

```
[root@web ~]# mkfs.ext4 /dev/vg02/lv02
```

挂载使用

```
[root@web ~]# mount /dev/mapper/vg02-lv02 /u02

[root@web ~]# touch /u02/file{1..3}
[root@web ~]# mkdir /u02/dir{1..3}
```

测试验证：

思路：损坏一个磁盘，测试数据是否在第二个物理卷中

1. 使用dd命令破坏一个物理卷

```
[root@web ~]# dd if=/dev/zero of=/dev/sdb7 bs=1M count=100
```

2. 再次查看物理卷发现有一个unknown Device

```
 /dev/sdc       vg03 lvm2 a--  20.00g 10.00g
unknown device vg02 lvm2 a-m   2.00g  4.00m
```

3. 将损坏的盘从卷组中移除

```
vgreduce vg02 --removemissing --force
```

4. 再次查看挂载点/u02数据依然存在

自己也可以再次测试：

1. 再拿刚刚人为损坏的盘做成物理卷再次加入到vg02卷组中

```
[root@web ~]# pvcreate /dev/sdb7
  Physical volume "/dev/sdb7" successfully created
[root@web ~]# vgextend vg02 /dev/sdb7
  Volume group "vg02" successfully extended
```

2. 再次让/dev/sdd5和/dev/sdd6互为镜像

```
[root@web ~]# lvconvert -m 1 /dev/vg02/lv02 /dev/sdb[78]
  vg02/lv02: Converted: 0.0%
  vg02/lv02: Converted: 32.2%
  vg02/lv02: Converted: 65.8%
  vg02/lv02: Converted: 97.7%
  vg02/lv02: Converted: 100.0%
```

3. 等待复制完成就可以再次人为模拟另一块物理卷损坏继续测试

四、逻辑卷快照

```
1. 创建快照 (EXT4)
[root@node1 ~]# lvcreate -L 128M -s -n lv2-snap /dev/vg1/lv2   给lv2逻辑卷创建快照
[root@node1 ~]# mount -o ro /dev/vg1/lv2-snap /mnt/lv2-snap/   挂载快照

[root@node1 ~]# lvscan     查看扫描快照
  ACTIVE            '/dev/vg1/lv1' [768.00 MiB] inherit
  ACTIVE   Original '/dev/vg1/lv2' [512.00 MiB] inherit
  ACTIVE   Snapshot '/dev/vg1/lv2-snap' [128.00 MiB] inherit

[root@node1 ~]# dmsetup ls --tree
vg1-lv2--snap (252:5)
 ├─vg1-lv2--snap-cow (252:7)     保存原卷改变前的数据
 |  └─ (253:17)
 └─vg1-lv2-real (252:6)          真实的逻辑卷（原卷）
    ├─ (253:17)
    └─ (253:18)
vg1-lv2 (252:1)
 └─vg1-lv2-real (252:6)
    ├─ (253:17)
    └─ (253:18)

2. 修改原卷的数据
[root@server ~]# dd if=/dev/zero of=/u01/test bs=1M count=30

3. 观察Snapshot
[root@server ~]# lvs /dev/vg1/lv2-snap
  LV     VG   Attr       LSize  Pool Origin Data%  Move Log Cpy%Sync Convert
  s-lv01 vg01 swi-aos--- 52.00m      lv01     0.16
[root@server ~]# lvs /dev/vg1/lv2-snap
  LV     VG   Attr       LSize  Pool Origin Data%  Move Log Cpy%Sync Convert
  s-lv01 vg01 swi-aos--- 52.00m      lv01    58.16


XFS:
[root@node1 ~]# mount -o nouuid,ro /dev/vg1/lv1-snap /mnt/lv1-snap/s
挂载快照，尽量使用ro的方式，将不会破坏快照卷中的数据

应用场景:
/var/lib/mysql
1. 锁表
2. 备份【逻辑|物理备份】
3. 解锁

100G  物理备份      /var/lib/mysql/xxx

1. 锁表
2. 创建快照
3. 解锁
4. 挂载快照
```

```
5. 备份到指定地方
6. 删除快照

快照实现自动扩容:
/etc/lvm/lvm.conf
snapshot_autoextend_threshold = 80
snapshot_autoextend_percent = 20
//当快照使用到80%时，自动扩容20%；当snapshot_autoextend_threshold = 100表示关闭自动扩容
```

五、磁盘配额

```
磁盘配额 quota
================================================================================
作用:  限制用户或组对磁盘空间的使用，例如文件服务器，邮件服务器...

一、启用磁盘限额
1. 让文件系统支持配额 [ext3/4]
# vim /etc/fstab
/dev/vg02/lv02          /u01            ext4    defaults,usrquota,grpquota 0 0
# umount /u01
# mount -a
# mount |grep u01
/dev/mapper/vg02-lv02 on /u01 type ext4 (rw,usrquota,grpquota)

2. 创建磁盘配额的数据库文件
注意:  建议停用SELinux
[root@server ~]# yum -y install quota
[root@server ~]# quotacheck -acug
[root@server ~]# ll /u01
total 16
-rw------- 1 root root 6144 Sep 17 09:28 aquota.group
-rw------- 1 root root 6144 Sep 17 09:28 aquota.user


//-a 所有分区（已支持配额）
//-c 创建
//-u 用户
//-g 组

3.启动磁盘配额
# quotaon -a          //启动所有分区的磁盘配额


二、日常管理
设置配额:
方法一: edquota
# edquota -u stu1
Disk quotas for user stu1 (uid 500):
  Filesystem                   blocks       soft       hard     inodes     soft     hard
  /dev/mapper/vg01-lv01             0          0          0          0        0        0
```

soft:又称软限制，当用户到达这个限制以后，系统会给予警告，但仍可写入。
hard:又称硬限制，到达这个限制，就完全禁止任何写入

以下三个为磁盘空间的限制设置：
blocks：已使用空间，无需要设置
soft：用户空间使用限制，为软限制，需要设置
hard：用户空间使用限制，为硬限制，需要设置
以下三个为总文件个数的限制：
inodes：已有文件总数，无需要设置
soft：文件总数限制，为软限制，需要设置
hard：文件总数限制，为硬限制，需要设置

我们要限制stu1用户使用空间10M，最多不能超过12M，文件总数为200个，
最多不能超过250个，设置如下：

```
Filesystem                blocks soft hard inodes soft hard
/dev/mapper/vg01-lv01       0     10240 12288 0    200  250
```

注：空间限制是以k为单位的。

grace time：宽限期，默认7天

```
# edquota -t  修改配额的宽限期
```

测试：

```
# su - stu1
[stu1@vm1 data]$ dd if=/dev/zero of=test99 bs=1M count=11
dm-1: warning, user block quota exceeded.
11+0 records in
11+0 records out
11534336 bytes (12 MB) copied, 0.108284 s, 107 MB/s


[stu1@vm1 data]$ dd if=/dev/zero of=test99 bs=1M count=13
dm-1: warning, user block quota exceeded.
dm-1: write failed, user block limit reached.
dd: writing `test99': Disk quota exceeded
13+0 records in
12+0 records out
12582912 bytes (13 MB) copied, 0.257964 s, 48.8 MB/s

[stu1@vm1 data]$ touch file{1..6}
dm-1: warning, user file quota exceeded.

[stu1@vm1 data]$ touch file{1..11}
dm-1: write failed, user file limit reached.
touch: cannot touch `file10': Disk quota exceeded
touch: cannot touch `file11': Disk quota exceeded

[stu1@vm1 data]$ quota     //查看自己的配额情况
Disk quotas for user stu1 (uid 500):
     Filesystem  blocks   quota   limit   grace   files   quota   limit   grace
/dev/mapper/vg01-lv01
                 12288*  10240   12288   24:00       1       5      10
```

方法二：setquota
```
# setquota -u username block软限制 block硬限制 inode软限制 inode硬限制 分区
# setquota -u jack 80000 100000 15 20 /dev/sda2
# quota jack
```

方法三：复制
```
# edquota -p alice tom robin user1 user2 user3
```
将alice的配额方案复制给后面所有用户

```
# for i in {1..10}
> do
> useradd zhang$i
> edquota -p stu1 zhang$i
> done
```

+++查看配额+++
查看单个用户的配额： # quota jack
查看所有用户的配额： # repquota -a
                                # repquota -ag
普通用户查看自己配额： $ quota

扩展知识：针对组设置配额
例1：限制hr组的成员能在/home/hr目录中：100M 50文件
```
# groupadd hr
# useradd hr01 -G hr
# useradd hr02 -G hr
# mkdir /home/hr
# chgrp hr /home/hr
# chmod 2770 /home/hr
# ll -d /home/hr
drwxrws--- 2 root hr 4096 09-12 17:07 /home/hr

# edquota -g hr
Disk quotas for group hr (gid 507):
   Filesystem                   blocks       soft       hard     inodes     soft
  hard
   /dev/mapper/vg01-lv_home          4          0     102400          1        0        50
# repquota -ag
================================================================================
```

rhel7：
注意：
1、不需要手动执行quotacheck命令对xfs文件系统进行检查，它会在mount的时候自动执行
2、不需要在xfs文件系统的根下生成quota数据文件

```
 # mount -o uquota /dev/xvm/home /home
 # xfs_quota -x -c 'limit bsoft=500m bhard=550m tanya' /home
 # xfs_quota -x -c report /home
```

```
-x：专家模式
-c：交互模式，可加多个
```

六、GPT分区

gdisk工具分区

parted工具分区

```
GPT  128个主分区
1.创建分区
# gdisk -l /dev/sdc
# gdisk /dev/sdc
# partprobe /dev/sdc
# ll /dev/sdc*

2.创建文件系统（格式化）redhat7默认使用xfs
# mkfs.xfs /dev/sdb1

3.挂载（手动、开机自动、autofs自动）
# mkdir /mnt/disk1
# mkdir /mnt/disk2
# mount -t xfs -o ro /dev/sdb1 /mnt/disk1    //手动挂载
# umount /mnt/disk1
```

课后实战

1. 基本练习

1. 添加一块新的10G物理磁盘到你的Linux操作系统中，并将其分为2个分区（大小自己决定）
2. 将sdb1做成大小为2G的逻辑卷lv01挂载到系统中的/u01目录下面，并且在/u01目录下面创建file1~file5 5个文件
3. 假设sdb1分区有坏块，现在需要将sdb1分区上的数据快速放到另外块盘sdb2上，怎么做?

```
1. 将/dev/sdb2加入到sdb1所在的卷组中
2. 使用pvmove命令移动
[root@server ~]# pvmove /dev/sdb1 /dev/sdb2
```

4. 由于业务需要，/u01目录需要扩大到9G，怎么做?
5. 新建一个卷组，名称为vg0，PEsize为8M，在卷组中创建一个名为lv02的逻辑卷，大小为50个PE，格式化为ext3，挂载在/mnt/lv01要求每次开机都生效。

```
[root@server ~]# vgcreate -s 8M vg0 /dev/sdb6
```

2. 实战演练

1. 部署好LAMP环境，MySQL数据库在本机的根文件系统中（如环境已有可以直接用）
2. 添加一块虚拟硬盘，使用LVM逻辑卷管理，将MySQL迁移到逻辑卷中，并保证网站正常访问