day04-linux磁盘管理-SWAP-LVM-RAID

- day04-linux磁盘管理-SWAP-LVM-RAID
  - 1.磁盘概述
    - 1.1 什么是磁盘
    - 1.2 磁盘物理结构
      - 1.2.1 什么是盘片
      - 1.2.2 什么是磁道
      - 1.2.3 什么是扇区
      - 1.2.4 什么是柱面
      - 1.2.5 什么是磁头
    - 1.3 磁盘的接口类型
      - 1.3.1 IDE-SCSI
      - 1.3.2 SATA-SAS
      - 1.3.3 MSATA-M2
  - 2.磁盘命名
    - 2.1 物理服务器
    - 2.2 虚拟服务器
  - 3.磁盘分区
    - 3.1 为什么要分区
    - 3.2 MBR分区
    - 3.3 GPT分区
  - 4.分区管理
    - 4.1 fdisk
    - 4.2 gdisk
    - 4.3 mkfs
  - 5.挂载管理
    - 5.1 临时挂载mount
    - 5.2 临时卸载umount
    - 5.3 永久挂载fstab
      - 5.3.1 永久挂载配置抒写
      - 5.3.2 配置文件/etc/fstab
  - 6.SWAP
    - 6.1 什么是SWAP
    - 6.2 为什么需要SWAP
    - 6.3 SWAP基本应用
      - 6.3.1 扩展swap分区

        - 6.3.2 缩小\swap分区
    - 7.文件系统修复
    - 8.磁盘阵列RAID
        - 8.1 什么是RAID
        - 8.2 为什么需要RAID
        - 8.3 实现RAID的几种模式
            - 8.3.1 RAID0
            - 8.3.2 RAID1
            - 8.3.3 RAID5
            - 8.3.4 RAID10
        - 8.4 实现RAID的方式
            - 8.4.1 硬RAID
            - 8.4.2 软RAID
        - 8.5 软RAID配置实战
            - 8.5.1 RAID环境准备
            - 4.5.1 RAID0实战
            - 4.5.2 RAID1实战
            - 4.5.3 RAID5实战
    - 9.逻辑卷lvm
        - 9.1 为何要用lvm
        - 9.2 什么是lvm
        - 9.3 lvm相关术语
        - 9.4 lvm配置实践
            - 9.4.1 环境与思路
            - 9.4.2 创建物理卷
            - 9.4.3 创建卷组
            - 9.4.4 创建逻辑卷
            - 9.4.5 挂载使用
        - 9.5 lvm卷组管理
            - 9.5.1 扩大卷组
            - 9.5.2 缩减卷组
        - 9.6 lvm逻辑卷管理
            - 9.6.1 扩展逻辑卷
            - 9.6.2 删除逻辑卷

> 徐亮伟，多年互联网运维工作经验，曾负责过大规模集群架构自动化运维管理工作。擅长Web集群架构与自动化运维，曾负责国内某大型电商运维工作。
> 个人博客"徐亮伟架构师之路"累计受益数万人。
> 笔者Q:552408925

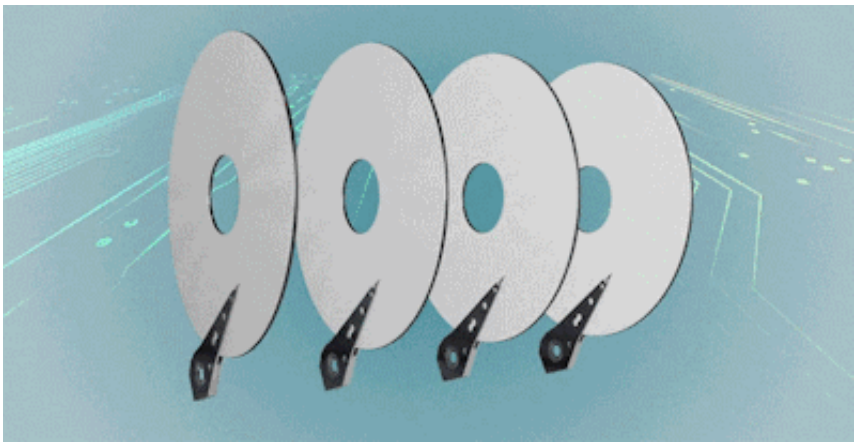
> 架构师群:471443208

1.磁盘概述

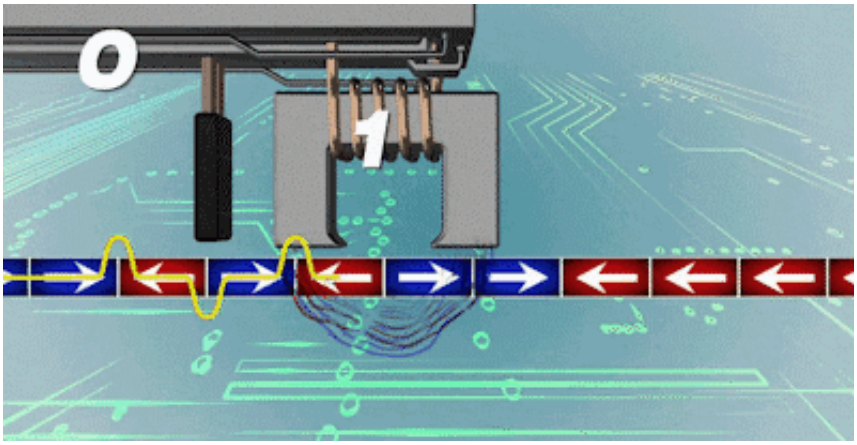
1.1 什么是磁盘

- 绝大多数人对硬盘都不陌生，
  - 一块小小的硬盘里，就可以存储海量的照片音乐和电影，尤其是我们喜爱的各类动作片。
  - 但如此小的空间，是如何储存那么多信息的呢?

每个硬盘中心都是一摞高速运转的圆盘，在圆盘上附着的一圈金属颗粒，每个金属颗粒都有自己的磁化程度，用于储存0和1。



当记录数据时，硬盘的磁头开始通电，形成强磁场，数据在磁场的作用下转变成电流，使颗粒磁化，从从而将信息记录在圆盘上。



由海量颗粒组成的信息，就是我们存在硬盘里的数据。

- 什么是磁盘、软盘、硬盘?

1.2 磁盘物理结构

- 磁盘物理结构-1点击此按钮
- 磁盘物理结构-2点击此按钮

1.2.1 什么是盘片

硬盘一般有一个或多个盘片，每个盘片可以有两面，即第一个盘片的正面为0面，反面为1面然后依次类推。

1.2.2 什么是磁道

每个盘片的盘面在出厂的时候被划分出了多个同心圆环，数据就存储在这样的同心圆环上面，我们将这样的圆环称为磁道 `Track`，每个盘面可以划分多个磁道。但肉业不可见。

1.2.3 什么是扇区

在硬盘出厂时会对磁盘进行一次低格，其实就是再每个磁道划分为若干个弧段，每个弧段就是一个扇区 `Sector`。扇区是硬盘上存储的物理单位，现在每个扇区可存储512字节数据已经成了业

界的约定。

1.2.4 什么是柱面

柱面实际上就是我们抽象出来的一个逻辑概念，简单来说就是处于同一个垂直区域的磁道称为 柱面 ，即各盘面上面相同位置磁道的集合。这样数据如果存储到相同半径磁道上的同一扇区，这样可以实现并行读取，主要是减少磁头寻道时间。

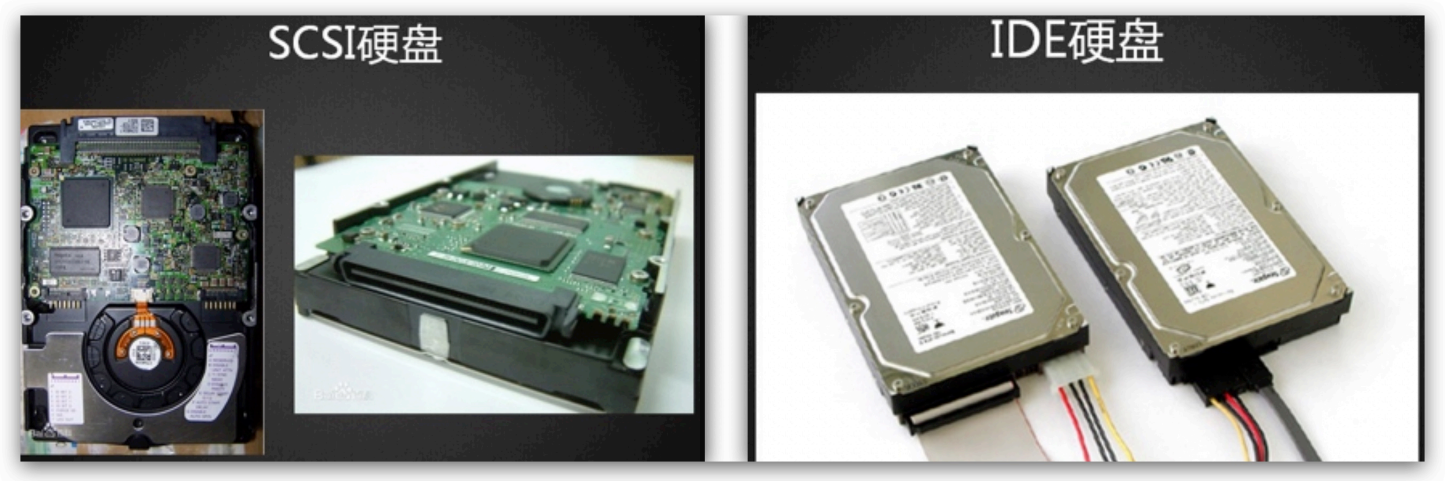
1.2.5 什么是磁头

读取磁盘磁道上面金属块，主要负责读或写入数据。

1.3 磁盘的接口类型

1.3.1 IDE-SCSI

`IDE, Scsi`（已经被淘汰）



1.3.2 SATA-SAS

- `SATA III` 与 `SAS`

接口类型	接口速率	盘片转速	写入速度	应用场景
SATA III	6Gbps/s	7.5k/s	300MB/s	个人
SAS	8Gbps/s~12Gbps/s	15k/s	300MB/s~600MB/s	企业

1.3.3 MSATA-M2

- `MSATA` 与 `M.2`

- `MSATA` 接口是专门为超级笔记本而设计的，m.2接口参考文档
- `m.2` 接口是 `inter` 推出的一种替代 `MSATA` 新的接口规范；
- `m.2` 接口相比 `MSATA` 接口有两方面的优势
  - 1.速度优势
  - 2.体积优势

M.2接口类型	支持接口类型	兼容性	读取速度	写入速度
Socket 2	SATA、PCI-E X2	几乎主板都支持	700MB/s	550MB/s
Socket 3	PCI-E ×4	需要检查主板是否支持	4GB/s	

2.磁盘命名

2.1 物理服务器

- 真实物理服务器

设备名称	分区信息	设备类型

/dev/sda	/dev/sda1	第一块物理磁盘第一分区
/dev/sdb	/dev/sdb2	第二块物理磁盘第二个分区
/dev/sdd	/dev/sdd4	第四块虚拟磁盘的第四个分区

2.2 虚拟服务器

- 阿里云主机或者KVM虚拟化主机的磁盘命名格式；

设备名称	分区信息	设备类型
/dev/vda	/dev/vda1	第一块虚拟磁盘的第一个分区
/dev/vdb	/dev/vdb2	第二块虚拟磁盘的第二个分区
/dev/vdc	/dev/vdc3	第三块虚拟磁盘的第三个分区

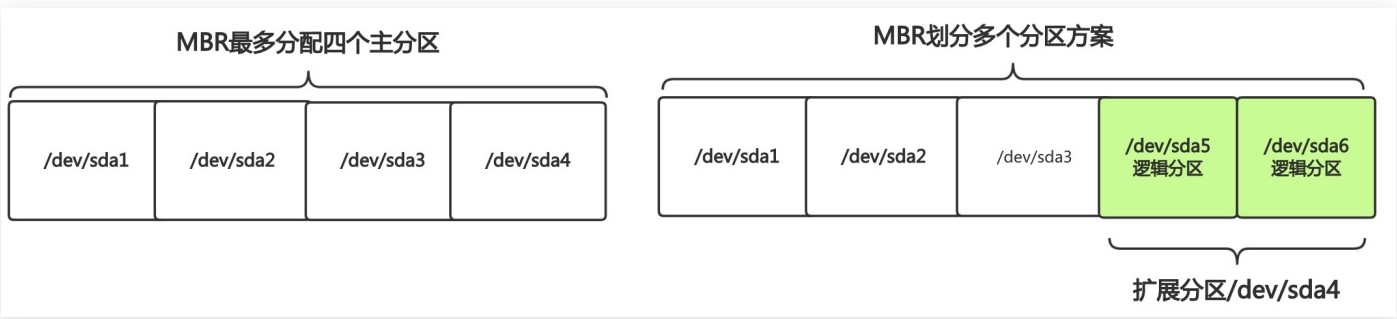
3.磁盘分区

3.1 为什么要分区

- 分区是为了便于数据分门别类的存储；分区有 `MBR、GPT` 两种方式；

3.2 MBR分区

- MBR分区仅能分配4个主分区，如果需要划分多个分区，需要先分配扩展分区，然后在分配逻辑分区
  - 1.为什么MBR分区仅能分配4个主分区？`MBR`只能划分4个主分区的原因，传送门
  - 2.分区编号如何表示：1~4主分区使用和扩展分区，逻辑分区从5开始



3.3 GPT分区

- 对于 `MBR` 分区而已，只能分配4个主分区，但新型的分区表 `GPT` 支持分配128个主分区。
  - `MBR` 与 `GPT` 有何区别，传送门
  - 注意 `MBR` 与 `GPT` 之间不能互转，会导致数据丢失。

4.分区管理

4.1 fdisk

- `fdisk` 仅支持分配小于 `2TB` 的磁盘
  - 查看当前设备 `fdisk -l`
  - 对设备进行分区 `fdisk /dev/sdb`
- 分区命令
  - m：显示帮助
  - n：创建新分区
  - d：删除分区
  - p：查看分区
  - w：保存分区
  - q：退出
- 分区案例：
  - 案例1：分配4个分区（4P）
  - 案例2：分配5个分区（1P+1E+4L）
  - 案例3：分配6个分区（3P+1E+3L）

4.2 gdisk

- `gdisk` 支持分配大于 `2TB` 的磁盘
  - 查看当前设备 `gdisk [-l] device`
  - 对设备进行分区 `gdisk /dev/sdb`
- 分区命令
  - ?：显示帮助
  - n：创建新分区
  - p：打印分区
  - w：保存分区
  - q：退出
- 分区案例：
  - 案例1：分配4个主分区（4P）
  - 案例2：分配5个主分区（5P）
  - 案例3：分配6个主分区（6P）

4.3 mkfs

- `mkfs` 命令用于格式化硬盘，类似于将房子装修成3室一厅，还是2室一厅；
  - `-b`：设定数据区块占用空间大小，目前支持 `1024、2048、4096 bytes` 每个块；
  - `-t`：用来指定什么类型的文件系统，可以是 `ext4、xfs`；
  - `-N`：设定 `inode` 数量，防止 `Inode` 数量不够导致磁盘不足；

1.使用 `mkfs` 命令，格式化整个硬盘

```
[root@xuliangwei ~]# mkfs.ext4  /dev/sdb
```

2.使用 `mkfs` 命令，格式化磁盘的某个分区

```
[root@xuliangwei ~]# mkfs.xfs  /dev/sdb1
```

5.挂载管理

- 当需要使用磁盘空间的时，需要准备一个目录作为挂载点，然后使用 `mount` 命令与该设备进行关联；

5.1 临时挂载mount

- 通过mount进行挂载，但重启将会失效。我们称为临时生效。
  - `-t`：指定文件系统挂载分区；
  - `-a`：检查并且挂载 `/etc/fstab` 配置文件中未挂载的设备；
  - `-o`：指定挂载参数，`ro、rw`；

1.挂载磁盘设备；

```
[root@xuliangwei ~]# mkdir /db1
[root@xuliangwei ~]# mount -t xfs /dev/sdb1  /db1
```

2.挂载磁盘设备，设置参数为仅可读；

```
[root@xuliangwei ~]# mkdir /db2
[root@dns-master ~]# mount -t xfs -o ro /dev/sdb2 /db2
[root@dns-master ~]# touch /db2/new_file
touch: cannot touch '/db2/new_file': Read-only file system
```

5.2 临时卸载umount

- 如果不想使用可以使用 `umount [device|directory]` 进行临时卸载。
  - `-l`：强制卸载；

1.卸载入口目录示例；

```
[root@xuliangwei ~]# umount /db1
```

2.卸载设备方式示例；

```
[root@xuliangwei ~]# umount /dev/sdb1
```

3.如碰到无法正常卸载情况处理；

```
[root@xuliangwei db1]# umount /db1
umount: /db1: device is busy.
        (In some cases useful info about processes that use
         the device is found by lsof(8) or fuser(1)

#如上情况解决办法有两种，1.切换至其他目录 2.强制卸载
[root@student db1]# umount -l /db1
```

5.3 永久挂载fstab

- 如果需要实现永久挂载，则需要将挂载相关信息写入 `/etc/fstab` 配置文件中实现。

5.3.1 永久挂载配置抒写

- 配置文件规范：`设备名称|挂载的入口目录|文件系统类型|挂载参数|是否备份|是否检查`
  - 1.获取设备名称，或者获取设备 `UUID`
  - 2.手动临时挂载测试；
  - 3.写入 `/etc/fstab` 配置文件；
  - 4.使用 `mount -a` 检查是否存在错误；

1.获取设备名称，或设备的`UUID`

```
[root@xuliangwei ~]# blkid |grep "sdb1"
/dev/sdb1: UUID="e271b5b2-b1ba-4b18-bde5-66e394fb02d9" TYPE="xfs"
```

2.手动挂载测试

```
[root@xuliangwei ~]# mount UUID="e271b5b2-b1ba-4b18-bde5-66e394fb02d9" /db1
```

3.写入 `/etc/fstab` 测试

```
# 手动编写
[root@xuliangwei ~]# tail -1 /etc/fstab
UUID=e271b5b2-b1ba-4b18-bde5-66e394fb02d9 /db1 xfs  defaults 0  0

# 自动实现
[root@xuliangwei ~]# blkid |grep /dev/sda1 | awk -F '[: ]+' '{print $2}' | \
sed -r 's#(.*)#\1 /db1 xfs defaults 0 0#g' >> /etc/fstab
```

4.加载 `/etc/fstab` 配置文件, 同时检测是否存在语法错误

```
[root@xuliangwei ~]# mount -a
```

5.3.2 配置文件/etc/fstab

- `/etc/fstab` 配置文件格式
  - 第一列：指定需要挂载的设备
    - 设备名称： `/dev/sdb1`
    - 设备ID： `UUID`
  - 第二列：挂载的入口目录
  - 第三列：文件系统类型
    - `xfs` 类型
    - `ext4` 类型
  - 第四列：挂载参数
    - `async/sync`： 使用同步或异步方式存储数据；默认 `async`
    - `user/nouser`： 是否允许普通用户使用mount命令挂载。默认 `nouser`
    - `exec/noexe`： 是否允许可执行文件执行。默认 `exec`
    - `suid/nosuid`： 是否允许存在 `suid` 属性的文件。默认 `suid`
    - `auto/noauto`： 执行 `mount -a` 命令时，此文件系统是否被主动挂载。默认 `auto`
    - `rw/ro`： 是否以只读或者读写模式进行挂载。默认 `rw`
    - `default`： 具有 `rw,suid,dev,exec,auto,nouser,async` 等参数；
  - 第五列：是否要备份磁盘
    - 0：不做备份

        - 1：每天进行备份操作
        - 2：不定日期的进行备份操作
    - 第六列： 开机是否检验扇区
        - 0：不要检验磁盘是否有坏道
        - 1：检验
        - 2：校验 (当1级别检验完成之后进行2级别检验)

6.SWAP

6.1 什么是SWAP

- `Swap` 分区在系统的物理内存不够时，将硬盘中的一部分空间供当前运行的程序使用。

6.2 为什么需要SWAP

- 当物理内存不够时会随机 `kill` 占用内存的进程，从而产生`oom`，临时使用 `swap` 可以解决。
- 案例：故障模拟；

```
[root@xuliangwei ~]# dd if=/dev/zero of=/dev/null bs=800M
#故障日志
[root@xuliangwei ~]# tail -f /var/log/messages
Out of memory: Kill process 2227 (dd) score 778 or sacrifice child
Killed process 2227 (dd) total-vm:906724kB, anon-rss:798820kB, file-rss:0kB
```

6.3 SWAP基本应用

1.创建分区，并格式化为 `swap` 分区。

```
[root@xuliangwei ~]# fdisk  /dev/sdb
# 格式化为swap
[root@xuliangwei ~]# mkswap /dev/sdb1
```

2.查看当前 `swap` 分区大小

```
[root@xuliangwei ~]# free -m
              total        used        free      shared  buff/cache   available
Mem:           1980        1475          80          10         424         242
```

```
Swap:          2047           4        2043
```

6.3.1 扩展swap分区

- 扩展 `swap` 分区，使用 `swapon` 命令
  - `swapon device`：将某个磁盘大小添加到 `swap` 分区中
  - `swapon -a`：添加所有有 `swap` 分区

```
[root@xuliangwei ~]# swapon /dev/sdb1
[root@xuliangwei ~]# free -m
              total        used        free      shared  buff/cache   available
Mem:           1980        1475          80          10         424         242
Swap:          3047           4        2043
```

6.3.2 缩小swap分区

- 缩小 `swap` 分区，使用 `swapoff` 命令
  - `swapoff device`：关闭某个磁盘的 `swap` 分区
  - `swapoff -a`：关闭所有 `swap` 分区

```
[root@xuliangwei ~]# swapoff /dev/sdb1
[root@xuliangwei ~]# free -m
              total        used        free      shared  buff/cache   available
Mem:           1980        1475          80          10         424         242
Swap:          2047           4        2043
```

7.文件系统修复

- 在 `Linux` 系统中，为了增加系统性能，通常系统会将一些数据先写入内存中，然后在刷新至磁盘中；
- 万一公司服务器突然断电或者其他未知原因，再次启动后，会造成文件系统错误；

```
fdisk /dev/sdc  #分配1G分区
mkfs.xfs /dev/sdc1
mount /dev/sdc1 /mnt
echo "Hello" > /mnt/new.txt

# 模拟损坏
dd if=/dev/zero of=/dev/sdc bs=300M count=1
umount /mnt
```

```
mount /dev/sdb1 /mnt #无法挂载

# 尝试修复
xfs_repair /dev/sdc1

# 强制修复
xfs_repair -L /dev/sdc1
```

8.磁盘阵列RAID

8.1 什么是RAID

- RAID简称磁盘阵列
- 什么是阵列，例如: 古代打仗时为什么要对士兵进行排兵布阵，其目的在于提高士兵整体的作战能力，而不是某个士兵的战斗力。
- 那么回到磁盘中，我们也可以将多块盘组合进行排列，提高磁盘的整体读写能力，和冗余能力，通常我们将其称为磁盘阵列。

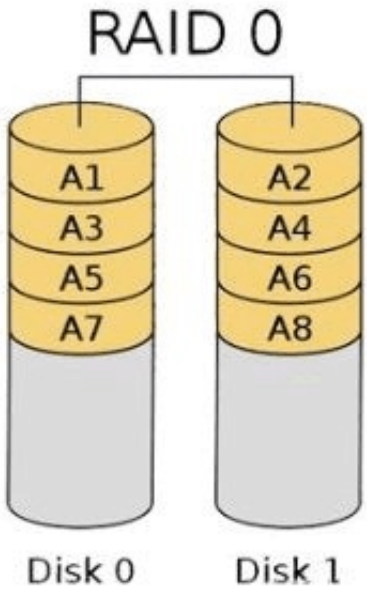
8.2 为什么需要RAID

- 1.提升读写能力。（在RAID中，可以让很多磁盘同时传输数据，因为多块磁盘在逻辑上感觉是一个磁盘，所以使用RAID可以达到单个磁盘的几倍、几十倍甚至上百倍的速率。）
- 2.保证数据安全。（硬盘其实非常的脆弱，它经常会坏掉。所以有了RAID这个东西。它的目的是将好几个硬盘合并在一起，就算硬盘坏了一个，剩下还有好几个硬盘是正常的，这样服务器还能正常提供服务。保证磁盘高可用。）
- RAID可以预防数据丢失，但并不能百分百保证数据不丢，所以在使用RAID的同时还是要备份重要的数据。

8.3 实现RAID的几种模式

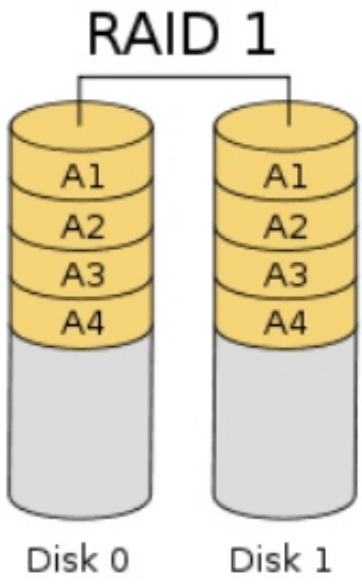
8.3.1 RAID0

- RAID0条带卷，最少两块盘。读写性能好，但没有容错机制。坏一块磁盘数据全丢。
  - 磁盘空间使用率：100%，成本低
  - 读性能：`N * 单块磁盘的读性能`；
  - 写性能：`N * 单块磁盘的写性能`；
  - 冗余：无，任何一块磁盘损坏都将导致数据不可用；
  - 应用场景：无状态服务（web）；



8.3.2 RAID1

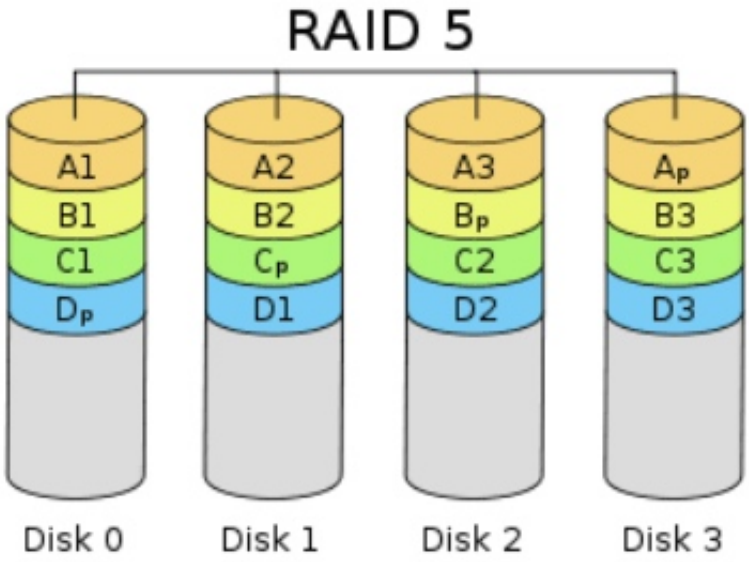
- `RAID1` 镜像卷，写入性能一般、读取性能快、有容错机制，但磁盘有50%浪费
  - 磁盘空间使用率：`50%` 成本较高。
  - 读性能：`N * 单块磁盘的读性能`；
  - 写性能：`1 * 单块磁盘的写性能`；
  - 冗余：在这一对镜像盘中有一块磁盘可以使用，那么无影响；
  - 应用场景：有状态服务（db）；



8.3.3 RAID5

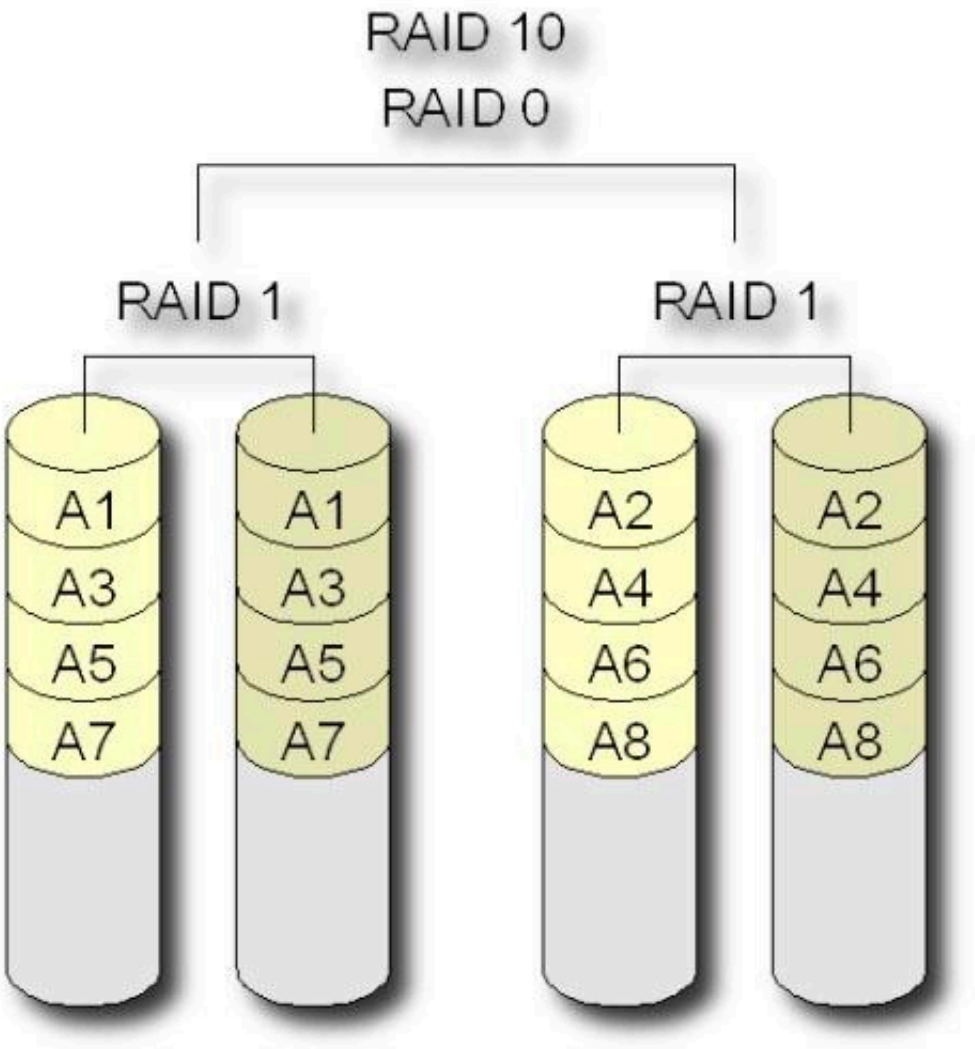
- `RAID5` 校验卷，至少3块相同大小的盘，并且只允许坏一块盘，有效空间1/3，读写速度快。坏掉一块盘，读会慢。
  - 磁盘空间利用率：`(N-1)`，即只浪费一块磁盘用于奇偶校验；

- 读性能：`(n-1)\*单块磁盘的读性能`，接近 `RAID0` 的读性能；
- 写性能：`(n-1)\*单块磁盘的写性能`，接近RAID0的写性能；
- 冗余：只允许一块磁盘损坏；
- 应用场景：常规选择（all）；



8.3.4 RAID10

- `RAID10`，先做`RAID1`，在做 `RAID0`
  - 磁盘空间利用率：`50%`
  - 读性能：
  - 写性能：
  - 冗余：只要一对镜像盘中有一块磁盘可以使用就没问题。
  - 应用场景：数据库（db）；



8.4 实现RAID的方式

8.4.1 硬RAID

- 硬 `RAID` 使用硬件阵列卡；在安装操作系统之前进入 `BIOS` 配置

8.4.2 软RAID

- 软 `RAID` 通过操作系统软件来实现，性能远不如硬 `RAID`， 仅测试效果；

8.5 软RAID配置实战

8.5.1 RAID环境准备

由于使用操作系统模拟的软RAID，所以需要在虚拟机上添加 `9` 块硬盘，来完成实验；

- 2.创建软 `RAID` 命令 `mdadm`，如果没有使用 `yum install mdadm` 安装即可
  - `mdadm` 磁盘阵列命令选项
  - 创建模式：
    - `-C`：创建阵列；
    - `-l`：指定指定级别；
    - `-n`：指定设备数量；
    - `-v`：指定设备名称；
    - `-x`：指定备用磁盘；
  - 管理模式:
    - --add
    - –remove
    - --fail

4.5.1 RAID0实战

- 创建 `RAID0` 实验环境：

raid种类	磁盘	热备盘
raid0	sdb、sdc	

1.创建 `raid0`

```
[root@xuliangwei ~]# mdadm  -C -v /dev/md0 -l 0 -n 2 /dev/sdb /dev/sdc
```

2.查看阵列信息

```
[root@xuliangwei ~]# mdadm -Ds
[root@xuliangwei ~]# mdadm -D /dev/md0
```

3.格式化磁盘并分区挂载

```
[root@xuliangwei ~]# mkfs.xfs /dev/md0
[root@xuliangwei ~]# mkdir /raid0
[root@xuliangwei ~]# mount /dev/md0 /raid0/
[root@xuliangwei ~]# df -h
```

4.5.2 RAID1实战

- 1）创建 `RAID1`，并添加1个热备盘；
- 2）模拟磁盘故障，看备用盘是否会自动顶替故障盘；
- 3）从 `raid1` 中移出故障盘；
- 创建 `RAID1` 实验环境：

raid种类	磁盘	热备盘
raid1	sdd、sde、	sdf

- 1.准备 `sdb、sdc` 两块盘，然后创建阵列为 `RAID1`，准备 `sdd` 为备用盘。

```
#1.创建raid1阵列
[root@xuliangwei ~]# mdadm -C -v /dev/md1 -l 1 -n 2 /dev/sd[d,e,f]
```

2.格式化磁盘并分区挂载；

```
[root@xuliangwei ~]# mkfs.xfs -f /dev/md1
[root@xuliangwei ~]# mkdir /raid1
[root@xuliangwei ~]# mount /dev/md1 /mnt/raid1/
```

3.使用 `--fail` 模拟 `RAID1` 中数据盘 `/dev/sde` 出现故障，观察 `/dev/sdf` 备用盘能否自动顶替故障盘；

```
[root@xuliangwei ~]# mdadm /dev/md1 --fail /dev/sde
```

4.检查当然 `raid` 状态

```
[root@xuliangwei ~]# # mdadm  -D /dev/md1
    Number   Major   Minor   RaidDevice State
       0       8       96        0      active sync   /dev/sdd
       2       8      128        1      spare rebuilding   /dev/sdf     # 热备盘已经
在同步数据
       1       8      112        -      faulty   /dev/sde   #故障盘
```

5.移除损坏的磁盘

```
[root@xuliangwei ~]# mdadm /dev/md1 -r /dev/sde
```

4.5.3 RAID5实战

- 1）使用三块盘创建 `RAID5`，使用 `-x` 添加热备盘
- 2）模拟损坏一块磁盘，然后查看备用盘是否能顶用 (此时是三块磁盘)
- 3）然后在模拟一块磁盘损坏，检查数据是否损坏 (此时是二块磁盘)
- 创建 `RAID5` 实验环境：

raid种类	磁盘	热备盘
raid5	sdg、sdh、sdi	sdj

1.创建 `raid5` 也可以在最后-x添加备用盘

```
[root@xuliangwei ~]# mdadm -C -v /dev/md5 -l 5 -n 3 /dev/sd{g,h,i,j}
```

2.格式化磁盘并分区挂载

```
[root@xuliangwei ~]# mkfs.xfs -f /dev/md5
[root@xuliangwei ~]# mkdir /mnt/raid5
[root@xuliangwei ~]# mount /dev/md5 /raid5/
[root@xuliangwei ~]# echo "Raid" > /raid5/file
[root@xuliangwei ~]# mdadm -D /dev/md5
```

3.模拟一块磁盘损坏，查看 `/dev/sdj` 备用磁盘是否会顶上

```
[root@xuliangwei ~]# mdadm /dev/md5 --fail /dev/sdg
[root@xuliangwei ~]# mdadm -D /dev/md5
```

4.将故障的 `/dev/sdg` 盘剔除；

```
[root@xuliangwei ~]# mdadm /dev/md5 -r /dev/sdg
```

5.再次模拟一块磁盘损坏，检查数据是否丢失；

```
[root@xuliangwei ~]# mdadm /dev/md5 --fail /dev/sdg
[root@xuliangwei ~]# mdadm -D /dev/md5
```

9.逻辑卷lvm

9.1 为何要用lvm

- 当刚开始安装Linux系统时，往往不能确定每个分区使用的空间大小，只能凭经验分配不科学；
  - 如果分区设置的过大，就浪费了磁盘空间；
  - 如果分区设置的过小，就会导致空间不够；
- 如何希望分配的空间过大或过小，都能动态调整，则需要使用到 `LVM` 逻辑卷；

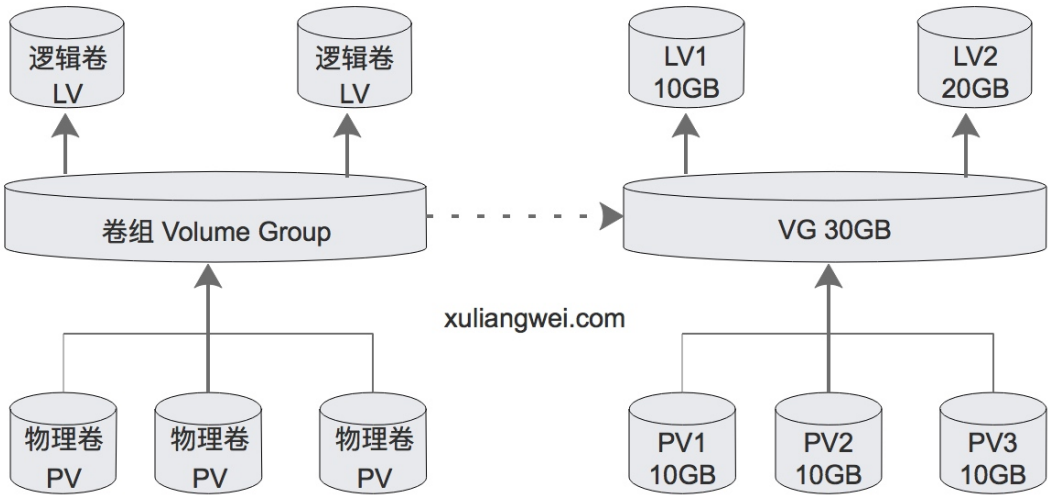
9.2 什么是lvm

- `LVM` 是 `Logical Volume Manager` 逻辑卷管理的简写，它是对磁盘分区管理的一种机制；
- `LVM` 优点：
  - `LVM` 可以创建和管理逻辑卷，而不是直接使用物理硬盘。
  - `LVM` 可以弹性的管理逻辑卷的扩大缩小，操作简单，而不损坏已存储的数据；
  - `LVM` 可以随意将新的硬盘添加到 `LVM`，以直接扩展已经存在的逻辑卷。
- `LVM` 缺点：
  - `LVM` 如果有一个磁盘损坏,整个 `lvm` 都坏了， `lvm` 只有动态扩展作用
  - 解决办法：用 `RAID + LVM` = 既有冗余又有动态扩展；

9.3 lvm相关术语

- 物理卷(PV)：将常规的磁盘通过 `pvcreate` 命令对其进行初始化，形成了物理卷。(面粉)
- 卷组(VG)：把多个物理卷组成一个逻辑的整体，这样卷组的大小就是多个盘之和。(大面团)
- 逻辑卷(LV)：从卷组中划分需要的空间大小出来，用户仅需对其格式化然后即可挂载使用。(切成馒头)

- 基本单元(PE)：分配的逻辑大小的最小单元，默认4MB，假设分配100MB的空间，则需要创建25个PE



9.4 lvm配置实践

9.4.1 环境与思路

- 1.准备三块物理磁盘，建议在虚拟机关闭状态添加，以便更好的实验；
- 1.创建物理卷，将普通磁盘转换为物理卷
- 2.创建卷组， 将物理卷加入到卷组中
- 3.在卷组中划分逻辑卷，然后挂载使用*

9.4.2 创建物理卷

1.将磁盘转换为物理卷，并加入 `pv`

```
[root@linux-node1 ~]# pvcreate /dev/sdb
Physical volume "/dev/sdb" successfully created.
```

2.检查 `pv` 创建情况

```
[root@linux-node1 ~]# pvs
PV         VG     Fmt  Attr PSize   PFree
/dev/sdb          lvm2 ---    1.00g 1.00g
```

9.4.3 创建卷组

1.创建名为 `datavg` 的卷组，然后将物理卷加入进卷组

```
[root@linux-node1 ~]# vgcreate datavg /dev/sdb
Volume group "datavg" successfully created
```

2.检查卷组(发现存在一个PV卷)

```
[root@linux-node1 ~]# vgs
VG      #PV #LV #SN Attr   VSize    VFree
datavg   1   0   0 wz--n- 1020.00m 1020.00m
```

9.4.4 创建逻辑卷

1.分配 `datavg` 逻辑卷， `-n` 指定逻辑卷名称， `-L` 指定逻辑卷大小；

```
#1.分配100M空间给lv1逻辑卷
[root@linux-node1 ~]# lvcreate -L 100M -n lv1 datavg
Logical volume "datalv1" created.
```

2.检查逻辑卷

```
[root@linux-node1 ~]# lvscan
ACTIVE            '/dev/datavg/lv1' [100.00 MiB] inherit
```

9.4.5 挂载使用

4.格式化逻辑卷，然后挂载使用。

1.格式化逻辑卷

```
[root@linux-node1 ~]# mkfs.xfs /dev/datavg/lv1
```

2.创建目录并挂载

```
[root@linux-node1 ~]# mkdir /lv1
[root@linux-node1 ~]# mount /dev/datavg/lv1 /lv1/
[root@linux-node1 ~]# df -h
Filesystem                 Size  Used Avail Use% Mounted on
...
/dev/mapper/datavg-lv1      97M  5.2M   92M   6% /lv1
```

9.5 lvm卷组管理

9.5.1 扩大卷组

1.准备新的磁盘加入至`pv`，然后检查卷组当前的大小；

```
[root@xuliangwei~]# pvcreate /dev/sdc
[root@xuliangwei ~]# vgs
  VG               #PV #LV #SN Attr   VSize    VFree
  datavg             1   1   0 wz--n- 1020.00m 920.00m
```

2.使用 `vgextend` 扩展卷组

```
[root@xuliangwei~]# vgextend datavg /dev/sdc
Volume group "datavg" successfully extended
```

3.再次检查，发现卷组已经扩大

```
[root@xuliangwei ~]# vgs
  VG               #PV #LV #SN Attr   VSize   VFree
  datavg             2   1   0 wz--n-   1.99g 1.89g
```

9.5.2 缩减卷组

- 假设想移除 `/dev/sdb` 磁盘，建议先将 `sdb` 磁盘数据先迁移到 `sdc` 磁盘，然后在移除；
- 注意：同一卷组的磁盘才可以进行在线迁移

1.检查当前逻辑卷 `VG` 中 `PV` 使用情况

```
[root@xuliangwei~]# pvs
PV      VG   Fmt  Attr PSize PFree
/dev/sdb vg1 lvm2 a -- 2.00g 1.76g
/dev/sdc vg1 lvm2 a -- 2.00g 2.00g
```

2. `pvmove` 在线数据迁移，将 `sdb` 的数据迁移至 `sdc`

```
[root@xuliangwei~]# pvmove /dev/sdb
  /dev/sdb: Moved: 100.00%
```

3.检查是否将 `sdb` 数据迁移至 `sdc`

```
[root@xuliangwei~]# pvs
PV       VG   Fmt  Attr PSize PFree
/dev/sdb vg1 lvm2 a -- 2.00g 2.00g
/dev/sdc vg1 lvm2 a -- 2.00g 1.76g
```

4.从卷组中移除 `sdb` 磁盘

```
[root@xuliangwei~]# vgreduce datavg /dev/sdb
  Removed "/dev/sdb" from volume group "datavg"
```

9.6 lvm逻辑卷管理

9.6.1 扩展逻辑卷

- 扩展逻辑卷：取决于 `vg` 卷中是否还有剩余的容量
- 注意扩展逻辑卷不能超过卷组 `VG` 的总大小

```
[root@xuliangwei~]# vgs
VG               #PV #LV #SN Attr   VSize    VFree
datavg             1   1   0 wz--n- 1020.00m 920.00m
```

1.扩展 `lv` 逻辑卷，增加 `800M` 分配给逻辑卷

```
[root@xuliangwei~]# lvextend -L +800M /dev/datavg/lv1

#也可以选择分配卷组中多少百分比给逻辑卷
[root@xuliangwei~]# lvextend -l +50%FREE /dev/datavg/lv1
```

2.扩展逻辑卷后需要更新fs文件系统

```
[root@xuliangwei~]# xfs_growfs /dev/datavg/lv1    #xfs文件格式扩容
[root@xuliangwei~]# resize2fs /dev/datavg/lv1     #ext文件格式扩容
```

9.6.2 删除逻辑卷

1.选卸载挂载点，然后在移除逻辑卷

```
[root@xuliangwei ~]# umount /dev/datavg/lv1
[root@xuliangwei ~]# lvremove /dev/datavg/lv1
```

2.删除 `vg`

```
[root@xuliangwei ~]# vgremove datavg
```

3.删除 `pv`

```
[root@xuliangwei ~]# pvremove /dev/sdb
[root@xuliangwei ~]# pvremove /dev/sdc
```