第4天-磁盘阵列/文件系统

一、开机挂载

```
[root@qfedu.com ~]#vim /etc/fstab
```

共6列

> 第1列:挂载设备（3种写法）
>
> ```
> (1)/dev/sda5
>
> (2)LABEL=卷标名称     rhel5的默认写法
>
> (3)UUID=设备的uuid  rhel6 rhel7的默认写法
> ```
>
> 第2列:挂载点
>
> 第3列:文件系统类型
>
> 第4列:文件系统属性
>
> 第5列:是否对文件系统进行(磁带)备份
>
> 0 不备份
>
> 1 1天一次
>
> 2 2天一次
>
> 第6列:是否检查文件系统
>
> 0 不检查
>
> 1 先检查
>
> 2 后检查

千锋云计算学院

二、开机挂载实战

实现磁盘分区的开机挂载并进行测试

三、磁盘阵列RAID

1、RAID介绍

RAID （Redundant Array of Independent Disks）即独立磁盘冗余阵列，通常简称为磁盘阵列。简单地说， RAID 是由多个独立的高性能磁盘驱动器组成的磁盘子系统，从而提供比单个磁盘更高的存储性能和数据冗余的技术。

核心概念

> 1、镜像（Mirroring）：镜像是一种冗余技术，为磁盘提供保护功能，防止磁盘发生故障而造成数据丢失。对于 RAID 而言，采用镜像技术 典型地 将会同时在阵列中产生两个完全相同的数据副本，分布在两个不同的磁盘驱动器组上。镜像提供了完全的数据冗余能力，当一个数据副本失效不可用时，外部系统仍可正常访问另一副本，不会对应用系统运行和性能产生影响。而且，镜像不需要额外的计算和校验，故障修复非常快，直接复制即可。镜像技术可以从多个副本进行并发读取数据，提供更高的读 I/O 性能，但不能并行写数据，写多个副本会会导致一定的 I/O 性能降低。
>
> 镜像技术提供了非常高的数据安全性，其代价也是非常昂贵的，需要至少双倍的存储空间。高成本限制了镜像的广泛应用，主要应用于至关重要的数据保护，这种场合下数据丢失会造成巨大的损失。另外，镜像通过“ 拆分 ”能获得特定时间点的上数据快照，从而可以实现一种备份窗口几乎为零的数据备份技术。
>
> 2、数据条带（Data Stripping）：磁盘存储的性能瓶颈在于磁头寻道定位，它是一种慢速机械运动，无法与高速的 CPU 匹配。再者，单个磁盘驱动器性能存在物理极限， I/O 性能非常有限。RAID 由多块磁盘组成，数据条带技术将数据以块的方式分布存储在多个磁盘中，从而可以对数据进行并发处理。这样写入和读取数据就可以在多个磁盘上同时进行，并发产生非常高的聚合 I/O ，有效提高了整体 I/O 性能，而且具有良好的线性扩展性。这对大容量数据尤其显著，如果不分块，数据只能按顺序存储在磁盘阵列的磁盘上，需要时再按顺序读取。而通过条带技术，可获得数倍于顺序访问的性能提升。
>
> 数据条带技术的分块大小选择非常关键。条带粒度可以是一个字节至几 KB 大小，分块越小，并行处理能力就越强，数据存取速度就越高，但同时就会增加块存取的随机性和块寻址时间。实际应用中，要根据数据特征和需求来选择合适的分块大小，在数据存取随机性和并发处理能力之间进行平衡，以争取尽可能高的整体性能。
> 数据条带是基于提高 I/O 性能而提出的，也就是说它只关注性能， 而对数据可靠性、可用性没有任何改善。实际上，其中任何一个数据条带损坏都会导致整个数据不可用，采用数据条带技术反而增加了数据发生丢失的概念率。
>
> 3、数据校验（Data parity）：镜像具有高安全性、高读性能，但冗余开销太昂贵。数据条带通过并发性来大幅提高性能，然而对数据安全性、可靠性未作考虑。数据校验是一种冗余技术，它用校验数据来提供数据的安全，可以检测数据错误，并在能力允许的前提下进行数据重构。相对镜像，数据校验大幅缩减了冗余开销，用较小的代价换取了极佳的数据完整性和可靠性。数据条带技术提供高性能，数据校验提供数据安全性， RAID 不同等级往往同时结合使用这两种技术。
>
> 采用数据校验时， RAID 要在写入数据同时进行校验计算，并将得到的校验数据存储在 RAID 成员磁盘中。校验数据可以集中保存在某个磁盘或分散存储在多个不同磁盘中，甚至校验数据也可以分块，不同 RAID 等级实现各不相同。当其中一部分数据出错时，就可以对剩余数据和校验数据进行反校验计算重建丢失的数据。校验技术相对于镜像技术的优势在于节省大量开销，但由于每次数据读写都要进行大量的校验运算，对计算机的运算速度要求很高，必须使用硬件 RAID 控制器。在数据重建恢复方面，检验技术比镜像技术复杂得多且慢得多

2、常用RAID分类及其区别

2.1、RAID 0

把连续的数据分散到多个磁盘上存取，系统有数据请求就可以被多个磁盘并行的执行，每个磁盘执行属于它自己的那部分数据请求。如果要做RAID 0，一台服务器至少需要两块硬盘，其读写速度是一块硬盘的两倍。如果有N块硬盘，那么读写速度就是一块硬盘的N倍。虽然读写速度可以提高，但由于没有数据备份功能，因此安全性会低很多。

优点：

> （1）充分利用I/O总线性能使其带宽翻倍，读/写速度翻倍。
>
> （2）充分利用磁盘空间，利用率为100%。

缺点：

> （1）不提供数据冗余。
>
> （2）无数据检验，不能保证数据的正确性。
>
> （3）存在单点故障。

应用场景：

> （1）对数据完整性要求不高的场景，如：日志存储，个人娱乐
>
> （2）要求读写效率高，安全性能要求不高，如图像工作站

2.2、RAID 1

通过磁盘数据镜像实现数据冗余，在成对的独立磁盘上产生互为备份的数据。当原始数据繁忙时，可直接从镜像拷贝中读取数据。同样地，要做RAID1也是至少需要两块硬盘，单读取数据时，一块会被读取，一块会被用作备份数据。其数据安全性就会较高，但是磁盘空间利用率是比较低的。

优点：

> （1）提供数据冗余，数据双倍存储。
>
> （2）提供良好的读性能

缺点：

> （1）无数据校验
>
> （2）磁盘利用率低，成本高

应用场景：

> 存放重要数据，如数据存储领域

2.3、RAID5

RAID5应该是目前最常见的 RAID 等级，它的原理与 RAID4 相似，区别在于校验数据分布在阵列中的所有磁盘上，而没有采用专门的校验磁盘。对于数据和校验数据，它们的写操作可以同时发生在完全不同的磁盘上。因此， RAID5 不存在 RAID4 中的并发写操作时的校验盘性能瓶颈问题。另外， RAID5 还具备很好的扩展性。当阵列磁盘 数量增加时，并行操作量的能力也随之增长，可比 RAID4 支持更多的磁盘，从而拥有更高的容量以及更高的性能。

RAID5的磁盘上同时存储数据和校验数据，数据块和对应的校验信息存保存在不同的磁盘上，当一个数据盘损坏时，系统可以根据同一条带的其他数据块和对应的校验数据来重建损坏的数据。与其他 RAID 等级一样，重建数据时， RAID5 的性能会受到较大的影响。

RAID5 兼顾存储性能、数据安全和存储成本等各方面因素，它可以理解为 RAID0 和 RAID1 的折中方案，是目前综合性能最佳的数据保护解决方案。 RAID5 基本上可以满足大部分的存储应用需求，数据中心大多采用它作为应用数据的保护方案。

优点：

> （1）读写性能高
>
> （2）有校验机制
>
> （3）磁盘空间利用率高

缺点：

> （1）磁盘越多安全性能越差

应用场景：

> 安全性高，如金融、数据库、存储等。

2.4、RAID6

前面所述的各个 RAID 等级都只能保护因单个磁盘失效而造成的数据丢失。如果两个磁盘同时发生故障，数据将无法恢复。RAID6引入双重校验的概念，它可以保护阵列中同时出现两个磁盘失效时，阵列仍能够继续工作，不会发生数据丢失。 RAID6 等级是在 RAID5 的基础上为了进一步增强数据保护而设计的一种 RAID 方式，它可以看作是一种扩展的 RAID5 等级。

RAID6 不仅要支持数据的恢复，还要支持校验数据的恢复，因此实现代价很高，控制器的设计也比其他等级更复杂、更昂贵。 RAID6 思想最常见的实现方式是采用两个独立的校验算法，假设称为 P 和 Q ，校验数据可以分别存储在两个不同的校验盘上，或者分散存储在所有成员磁盘中。当两个磁盘同时失效时，即可通过求解两元方程来重建两个磁盘上的数据。

RAID6 具有快速的读取性能、更高的容错能力。但是，它的成本要高于 RAID5 许多，写性能也较差，并有设计和实施非常复杂。因此， RAID6 很少得到实际应用，主要用于对数据安全等级要求非常高的场合。它一般是替代 RAID10 方案的经济性选择。

优点：

> （1）良好的随机读性能
>
> （2）有校验机制

缺点：

> （1）写入速度差
>
> （2）成本高

应用场景：

> 对数据安全级别要求比较高的企业

2.5、RAID10

先做镜像再作条带化

优点：

> （1）RAID10的读性能将优于RAID01
>
> （2）较高的IO性能
>
> （3）有数据冗余
>
> （4）无单点故障
>
> （5）安全性能高

缺点：

> 成本稍高

应用场景：

> 特别适用于既有大量数据需要存取，同时又对数据安全性要求严格的领域，如银行、金融、商业超市、仓储库房、各种档案管理等。

特例：
raid可以用动图介绍

千锋云计算学院

四、RAID实战

在【DELL服务器】上设置RAID

五、文件系统

1、常见文件系统类型

> ext3
>
> ext4
>
> xfs
>
> ntfs
>
> fat32

2、xfs文件系统修复

xfs_repair

主要用来修复xfs文件系统出现的损坏错误，一些选项含义如下：

> -L 强制日志文件(log)清空。日志文件中包含了一些元数据的修改，加上这个选项会强制清空log。这个选项会可能导致用户文件丢失。xfs是一个日志文件系统，对于日志文件系统不了解的可以查看文末的链接进行阅读。

> -n No modify mode.不修改，只对文件系统进行检查，查看需要那种类型的修复。

xfs_check 以及 xfs_repair -n都可以对xfs文件系统进行检查，但是xfs_check需要很长的时间。

xfs_repair结束后，所有不能到达的inode节点被放进lost+found文件夹。

使用xfs_repair对文件系统进行修复

使用xfs_repair之前，要先卸载该文件系统，然后使用xfs_metadump保存元数据信息

```
[root@qfedu.com ~]# xfs_metadump -o /dev/sdb1 /save/sdb1.metadump
```

这步只是对元数据进行了保存，保存在了/save/sdb1.metadump这个文件，没有对数据进行备份，因为如果要备份数据本身的话，需要和数据一样大小的空间，显然这里没有这个空间。

检查:

```
[root@qfedu.com ~]# xfs_repair -n /dev/sdb1
```

会进行一系列的检查。完成之后，进行下面的修复。

修复

```
[root@qfedu.com ~]# xfs_repair  /dev/sdb1
```

出现下面的错误:

```
ERROR: The filesystem has valuable metadata changes in a log which needs to be
replayed.  Mount the filesystem to replay the log, and unmount it before re-
running xfs_repair.  If you are unable to mount the filesystem, then use the -
L option to destroy the log and attempt a repair. Note that destroying the log
may cause corruption -- please attempt a mount of the filesystem before doing
this.
```

这个时候，因为之前已经无法挂载了，所以也就不能进行replay的操作了，因此只能进行使用-L选项了。

```
[root@qfedu.com ~]# xfs_repair -L /dev/sdb1
```

会进行log的清空，然后一系列的检查，得到下面的结果:

```
...
Phase 4 - check for duplicate blocks...
        - setting up duplicate extent list...
        - check for inodes claiming duplicate blocks...
        - agno = 0
        - agno = 4
        - agno = 1
        - agno = 8
```

千锋云计算学院

```
        - agno = 9
        - agno = 10
        - agno = 11
        - agno = 12
        ...
Phase 5 - rebuild AG headers and trees...
        - reset superblock...
Phase 6 - check inode connectivity...
        - resetting contents of realtime bitmap and summary inodes
        - traversing filesystem ...
        - traversal finished ...
        - moving disconnected inodes to lost+found ...
```

再执行修复

```
[root@qfedu.com ~]# xfs_repair /dev/sdb1
```

执行完后结果,有省略:

```
Phase 1 - find and verify superblock...
Phase 2 - using internal log
        - zero log...
        - scan filesystem freespace and inode maps...
        - found root inode chunk
Phase 3 - for each AG...
        - scan and clear agi unlinked lists...
        - process known inodes and perform inode discovery...
        - agno = 0
        - agno = 1
        - agno = 2
        - agno = 3
        - agno = 4
        - agno = 5
        ...
        - process newly discovered inodes...
Phase 4 - check for duplicate blocks...
        - setting up duplicate extent list...
        - check for inodes claiming duplicate blocks...
        - agno = 0
        - agno = 1
        - agno = 2
        - agno = 3
        - agno = 4
        - agno = 5
        - agno = 7
        - agno = 8
        ...
Phase 5 - rebuild AG headers and trees...
```

千锋云计算学院

```
        - reset superblock...
Phase 6 - check inode connectivity...
        - resetting contents of realtime bitmap and summary inodes
        - traversing filesystem ...
        - traversal finished ...
        - moving disconnected inodes to lost+found ...
Phase 7 - verify and correct link counts...
done
```

条件时间允许的话，可以再进行个检查，但是考虑到时间太久，这里就不执行最后的检查了。xfs_check耗时较长。

```
[root@qfedu.com ~]# xfs_check /dev/sdb1
```

这个时候，进去看原来的/data文件夹的数据，已经出来了。因此可以进行挂载了。

3、理解inode

> 理解inode，要从文件储存说起。
>
> 文件储存在硬盘上，硬盘的最小存储单位叫做"扇区"（Sector）。每个扇区储存512字节（相当于0.5KB）。
>
> 操作系统读取硬盘的时候，不会一个个扇区地读取，这样效率太低，而是一次性连续读取多个扇区，即一次性读取一个"块"（block）。这种由多个扇区组成的"块"，是文件存取的最小单位。"块"的大小，最常见的是4KB，即连续八个 sector组成一个 block。
>
> 文件数据都储存在"块"中，那么很显然，我们还必须找到一个地方储存文件的元信息，比如文件的创建者、文件的创建日期、文件的大小等等。这种储存文件元信息的区域就叫做inode，中文译名为"索引节点"。
>
> 每一个文件都有对应的inode，里面包含了与该文件有关的一些信息。

inode的内容

> 文件的字节数
>
> 文件拥有者的User ID
>
> 文件的Group ID
>
> 文件的读、写、执行权限
>
> 文件的时间戳，共有三个：ctime指inode上一次变动的时间，mtime指文件内容上一次变动的时间，atime指文件上一次打开的时间。
>
> 链接数，即有多少文件名指向这个inode
>
> 文件数据block的位置

4、文件链接

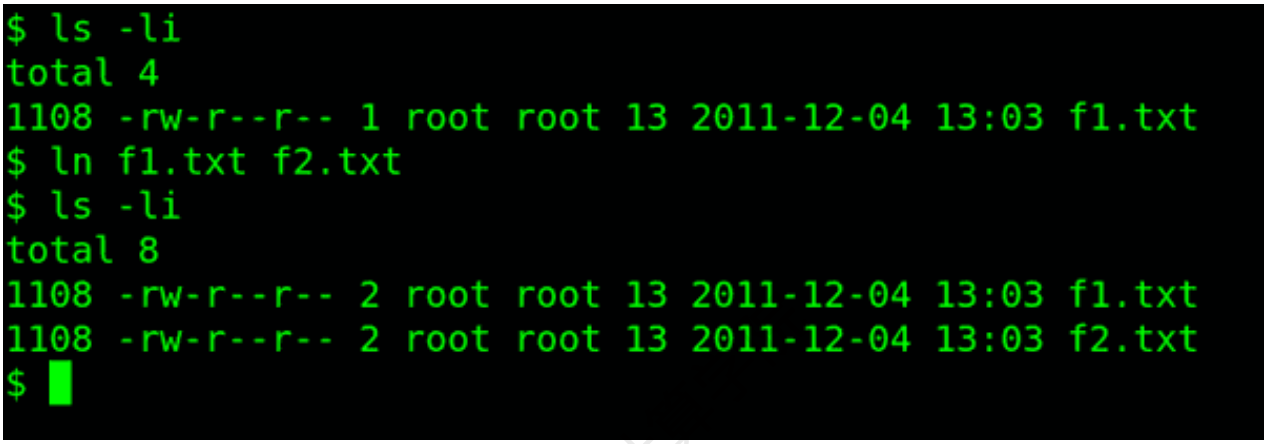
4.1、硬链接

一般情况下，文件名和inode号码是"一一对应"关系，每个inode号码对应一个文件名。但是，Unix/Linux系统允许，多个文件名指向同一个inode号码。

这意味着，可以用不同的文件名访问同样的内容；对文件内容进行修改，会影响到所有文件名；但是，删除一个文件名，不影响另一个文件名的访问。这种情况就被称为"硬链接"（hard link）。

ln命令可以创建硬链接：

```
[root@qfedu.com ~]# ln 源文件 目标文件
```



> 运行上面这条命令以后，源文件与目标文件的inode号码相同，都指向同一个inode。inode信息中有一项叫做"链接数"，记录指向该inode的文件名总数，这时就会增加1。
>
> 反过来，删除一个文件名，就会使得inode节点中的"链接数"减1。当这个值减到0，表明没有文件名指向这个inode，系统就会回收这个inode号码，以及其所对应block区域。
>
> 这里顺便说一下目录文件的"链接数"。创建目录时，默认会生成两个目录项："."和".."。前者的inode号码就是当前目录的inode号码，等同于当前目录的"硬链接"；后者的inode号码就是当前目录的父目录的inode号码，等同于父目录的"硬链接"。所以，任何一个目录的"硬链接"总数，总是等于2加上它的子目录总数（含隐藏目录）。

4.2、软链接

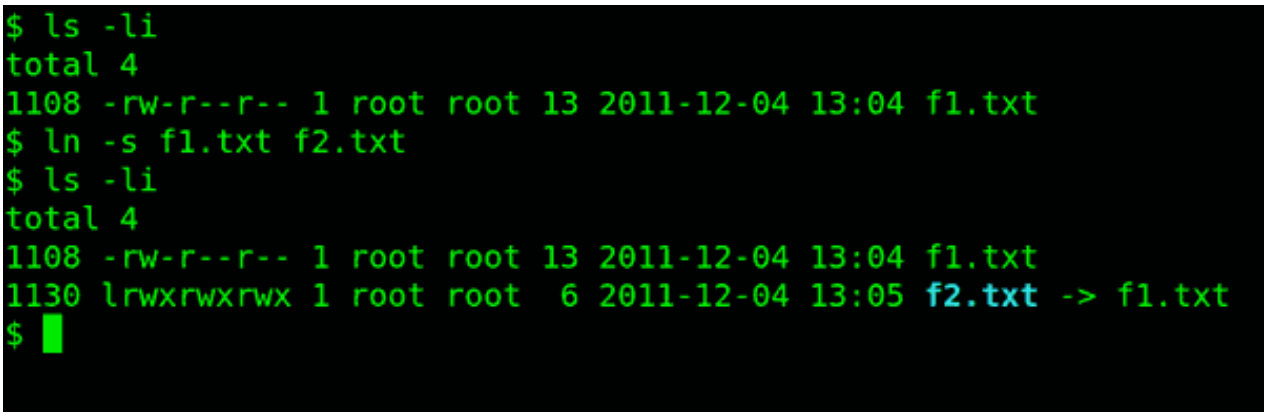
除了硬链接以外，还有一种特殊情况。

文件A和文件B的inode号码虽然不一样，但是文件A的内容是文件B的路径。读取文件A时，系统会自动将访问者导向文件B。因此，无论打开哪一个文件，最终读取的都是文件B。这时，文件A就称为文件B的"软链接"（soft link）或者"符号链接（symbolic link）。

这意味着，文件A依赖于文件B而存在，如果删除了文件B，打开文件A就会报错："No such file or directory"。这是软链接与硬链接最大的不同：文件A指向文件B的文件名，而不是文件B的inode号码，文件B的inode"链接数"不会因此发生变化。

ln -s 命令可以创建软链接。

```
[root@qfedu.com ~]# ln -s 源文件或目录 目标文件或目录
```



六、实战

为一个文件创建软连接和硬连接并测试其特性

千锋云计算学院