

操作系统存储管理手册

1. Linux.....	4
1.1. 文件系统的概述.....	7
1.1.1. 术语.....	7
1.1.2. Linux 中的主要文件系统.....	7
1.2. 使用 UUID 装入设备.....	9
1.2.1. 了解 UUID.....	9
1.2.2. 查找文件系统设备的 UUID.....	9
1.3. 使用 UDEV 进行设备绑定.....	11
1.3.1. udev 工作流程图.....	11
1.3.2. 磁盘绑定块设备.....	11
1.3.3. 磁盘绑定裸设备.....	12
1.3.4. 磁盘分区绑定块设备 (RHEL5).....	12
1.3.5. 磁盘分区绑定块设备 (RHEL6).....	14
1.3.6. 磁盘分区绑定裸设备.....	16
1.3.7. UDEV 的值和可调用的替换操作符.....	17
1.3.8. UDEV 常用命令.....	17
1.4. 管理设备的多路径 I/O.....	19
1.4.1. 规划多路径.....	20
1.4.2. 多路径管理工具.....	22
1.4.3. 为多路径配置系统.....	23
1.4.4. 创建或修改 /etc/multipath.conf 文件.....	23
1.4.5. 将非多路径设备列入黑名单.....	24
1.4.6. 多路径设备名称类型.....	25
1.4.7. 在不重引导的情况下扫描新设备.....	25
1.4.8. 查看多路径 I/O 状态.....	25
1.5. 软件 RAID 配置.....	26
1.5.1. 了解 RAID 级别.....	28
1.5.2. 查错软件 RAID.....	28
1.5.3. 创建 RAID.....	28
1.5.4. 嵌套的 RAID 设备.....	29
1.6. 经由 IP 网络的大容量储存: iSCSI.....	30
1.7. 以太网光纤通道储存: FCoE.....	30
1.8. 磁盘管理常用命令.....	30
1.8.1. blkid.....	30
1.8.2. /etc/fstab.....	31
1.9. 磁盘管理常用文件.....	32
1.9.1. /proc/partitions.....	32

2. LVM 配置.....	34
2.1. LVM 基本概念.....	34
2.2. 创建 LVM 分区.....	35
2.3. 配置物理卷 PV.....	36
2.4. 创建卷组 VG.....	36
2.5. 配置逻辑卷 LV.....	37
2.6. 调整文件系统大小.....	37
2.7. 常用命令.....	38
2.7.1. 扫描块设备 lvmdiskscan.....	38
2.7.2. 物理卷.....	38
2.7.3. 卷组.....	39
2.7.4. 逻辑卷.....	39
2.7.5. 检查元数据.....	40
2.7.6. e2fsck.....	40
2.7.7. resize2fs.....	40
3. Veritas Volume Manager.....	41
3.1. Veritas Volume Manager 基本概念.....	41
3.1.1. VxVM 和操作系统.....	41
3.1.2. VxVM 如何进行存储管理.....	41
3.2. 管理磁盘.....	46
3.2.1. 磁盘设备.....	46
3.2.2. 发现和配置新添加的磁盘设备.....	47
3.2.3. 发现磁盘和动态添加磁盘阵列.....	48
3.2.4. 将磁盘置于 VxVM 控制之下.....	48
3.3. 常用命令.....	49
3.3.1. 获取关于 VxVM 对象的信息.....	49
3.3.2. 管理磁盘.....	49
3.3.3. 创建和管理磁盘组.....	49
3.3.4. 创建和管理子磁盘.....	49
3.3.5. 创建和管理 Plex.....	49
3.3.6. 创建卷.....	49
3.3.7. 监视和控制任务.....	49
4. 5.....	50

1. 存储知识基础

1.1. DAS

直接连接存储 (**DAS:Direct Attached Storage**), 是指将存储设备通过 **SCSI** 接口或 **FC** 接口直接连接到一台计算机上。**DAS** 不算网络存储, 因为只有它所挂载的主机才可访问它。

直连式存储与服务器主机之间的连接通道通常采用 **SAS** 连接, 随着服务器 **CPU** 的处理能力越来越强, 存储硬盘空间越来越大, 阵列的硬盘数量越来越多, **SAS** 通道将会成为 **IO** 瓶颈; 服务器主机 **SAS ID** 资源有限, 能够建立的 **SAS** 通道连接有限。

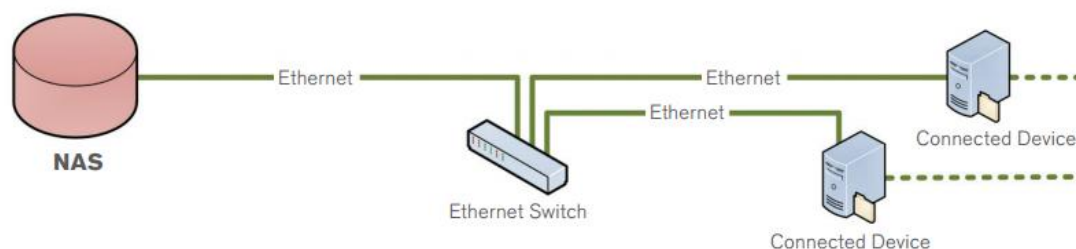
1.2. NAS

网络连接存储 (**NAS:Network Attached Storage**), 是指将存储设备通过标准的网络拓扑结构 (例如以太网), 连接到一群计算机上。**NAS** 有文件系统和 **IP** 地址, 可以类似的理解为网上邻居的共享磁盘。

NAS 是文件级的存储方法, 它的重点在于帮助工作组和部门级机构解决迅速增加存储容量的需求。如今用户采用 **NAS** 较多的功能是用来文档共享、图片共享、电影共享等等, 而且随着云计算的发展, 一些 **NAS** 厂商也推出了云存储功能, 大大方便了企业和个人用户的使用。

NAS 产品是真正即插即用的产品。**NAS** 设备一般支持多计算机平台, 用户通过网络支持协议可进入相同的文档, 因而 **NAS** 设备无需改造即可用于混合 **Unix/windows** 局域网内, 同时 **NAS** 的应用非常灵活。

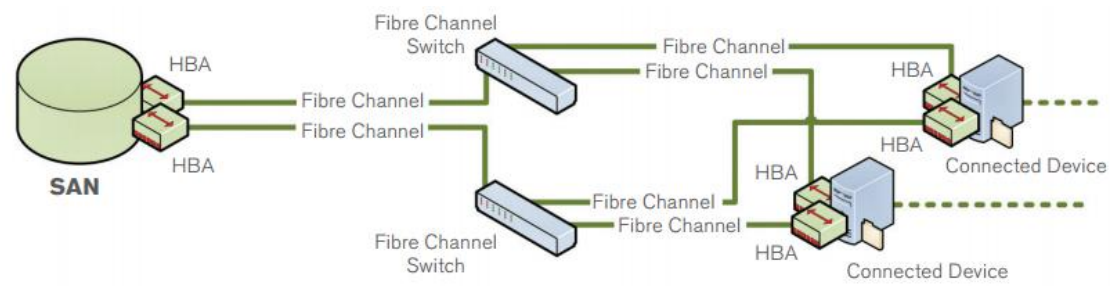
但 **NAS** 有一个关键性问题, 即备份过程中的带宽消耗。与将备份数据流从 **LAN** 中转移出去的存储区域网 (**SAN**) 不同, **NAS** 仍使用网络进行备份和恢复。**NAS** 的一个缺点是它将存储事务由并行 **SCSI** 连接转移到了网络上。这就是说 **LAN** 除了必须处理正常的最终用户传输流外, 还必须处理包括备份操作的存储磁盘请求。**NAS** 性能比 **SAN** 要差很多, 但相对成本也低很多。



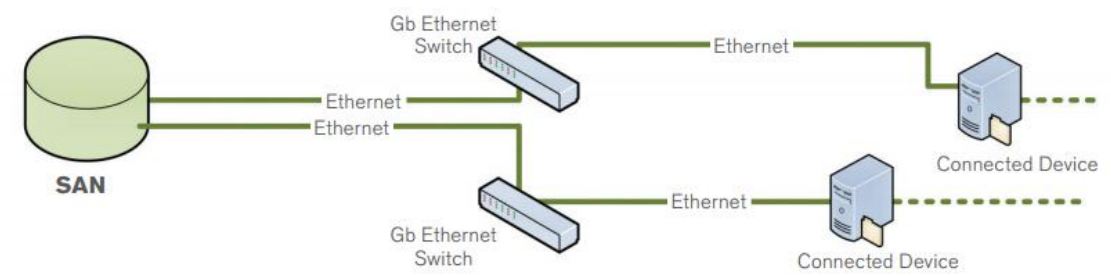
1.3. SAN

存储区域网 (**SAN: Storage Area Network**), 目前的 **SAN** 存储有 2 种: 一是基于光纤通道的 **FC SAN**; 二是基于以太网的 **IP SAN** (也就常说的 **iSCSI**)。

FC SAN 通过光纤交换机连接到主机 (**HBA** 卡), 也就是说可以连接到光纤交换机的主机都可以访问这个存储;



iSCSI 作为共享于以太网上的存储则更类似于 NAS。



存储区域网络，从名字上我们也可以看出，这个是通过光纤通道交换机连接存储阵列和服务端主机，最后成为一个专用的存储网络。

1.4. 各类存储对比

存储类型	DAS	NAS	iSCSI/IP SANs	光纤通道
价格	价格较低	价格中等	价格中等到较高	价格较高
可扩展性	非常有限	依赖于解决方案	依赖于解决方案	依赖于解决方案
可管理性	效率较低	效率较低	非常高效	非常高效
容错性	一定程度的容错性	一定程度的容错性	容错性很好	容错性很好
是否适合文件存储	是	是	是	是
是否适合数据库存储	是	否	通常适合	是
是否适合网页服务	是	是	是	是
是否适合 Exchange 存储	是	否	通常适合	是
安装的简易性	简单	简单	有一定的困难	非常困难
灾难恢复的能力	没有	没有	很多	很多

操作系统的支持	全部	N/A	windows, Linux, UNIX, NetWare	windows, Linux, UNIX, NetWare
主要提供商	任何服务器提供商	Huawei、IBM, Dell, HP, Network Appliance	Huawei、 LeftHand, EMC, HP, IBM, Network, Appliance	Huawei、IBM, EMC, HP, Network Appliance

2. Linux

2.1. 文件系统的概述

2.1.1. 术语

2.1.1.1 元数据

文件系统内部的一种数据结构。它确保能够正确组织和访问所有磁盘上的数据。

2.1.1.2 inode

文件系统上的一种数据结构，包含关于文件的各种信息，包括大小、链接数、指向实际储存文件内容的磁盘块的指针以及创建、修改和访问的日期与时间。

2.1.1.3 日记

在提及文件系统时，日记是包含某种日志的磁盘上结构，文件系统将要对文件系统的元数据进行的更改储存在此日志中。

2.1.2. Linux 中的主要文件系统

记住一点很重要：没有任何一种文件系统适合所有种类的应用。每个文件系统都有各自的特定优点和缺点，必须将这些因素考虑在内。

2.1.2.1 Btrfs

要将 **Ext** 文件系统转换为 **Btrfs** 文件系统，请将文件系统脱机，然后输入：

```
btrfs-convert <device>
```

要将迁移回滚到原先的 **Ext** 文件系统，请将文件系统脱机，然后输入：

```
btrfs-convert -r <device>
```

2.1.2.2 Ext2

在系统中断后，如果无法彻底卸装文件系统，则 **e2fsck** 将开始分析文件系统数据。系统使元数据恢复一致的状态，并将挂起的文件或数据块写入指定的目录（名为 **lost+found**）。

2.1.2.3 Ext3

inode 的数量控制着文件系统中可容纳的文件数：一个文件对应一个 **inode**。

重要

inode 分配完毕后，将无法更改 **inode** 大小的设置或每 **inode** 的字节数比率。除非使用不同的设置重新创建文件系统或扩展文件系统，否则将无法创建新的 **inode**。超过 **inode** 最大数量时，只有删除部分文件才能在文件系统上创建新文件。

inode 大小： 默认的 **inode** 大小为 256 字节。指定字节值，即介于 128 字节（含）到块大小（含）之间的 2 的乘方值，如 128、256、512，以此类推。

每 inode 的字节数比率： 默认的每 **inode** 的字节数比率为 16384 字节。有效的每 **inode** 的字节数比率值必须是大于等于 1024 字节的 2 的乘方值，如 1024、2048、4096、8192、16384、32768，以此类推。

```
[root@oeldb1 ~]# ls -la /etc/mke*fs.*
-rw-r--r-- 1 root root 330 Oct 12 2012 /etc/mke2fs.conf
-rw-r--r-- 1 root root 801 Sep 8 2009 /etc/mke4fs.conf
```

```
mkfs.ext3 -b 4096 -i 8092 -I 128 /dev/sda2
mke2fs -t ext3 -b 4096 -i 8192 -I 128 /dev/sda2
```

2.1.2.4 ReiserFS

2.1.2.5 XFS

SGI 在 20 世纪 90 年代初开始开发 **XFS**，最初计划将 **XFS** 作为 **IRIX OS** 的文件系统。开发 **XFS** 的目的是创建一个高性能的 64 位日记文件系统来满足对计算能力的极高要求。**XFS** 适合操纵大型文件，在高端硬件上表现优异。但即使是 **XFS** 也有缺点。与 **ReiserFS** 类似，**XFS** 非常注重元数据的完整性，但不太注重数据的完整性。

2.2. 使用 UUID 装入设备

在 Linux 2.6 和更新的内核中，udev 使用永久性设备命名，为动态的 /dev 目录提供了一种用户空间解决方案。作为热插拔系统的一部分，在系统中添加或删除设备时会执行 udev。

使用一个规则列表来针对特定设备属性进行匹配。udev 规则基础设施（在 /etc/udev/rules.d 目录中定义）为所有磁盘设备提供了稳定的名称，无论它们的识别顺序或该设备使用的连接如何。

2.2.1. 了解 UUID

UUID（全球唯一标识符）是表示文件系统的 128 位数字，在本地系统和其他系统中都是唯一的。它是使用系统硬件信息和时戳作为其源的一部分随机生成的。UUID 通常用于唯一性标记设备。

UUID 在分区中总是唯一的，不取决于它出现的顺序或装入的位置。当某些 SAN 设备挂接到服务器时，系统分区会重命名并移到最后一个设备。例如，如果在安装期间将 root (/) 指派给 /dev/sda1，则在连接 SAN 之后它会被指派给 /dev/sdg1。避免此问题的一个方法是在引导加载程序和引导设备的 /etc/fstab 文件中使用 UUID。

无论设备的装入位置如何，制造商为驱动器指派的设备 ID 不会更改，所以总是可以在引导时找到它。UUID 是文件系统的属性，如果重新格式化驱动器，UUID 会更改。

2.2.2. 查找文件系统设备的 UUID

在 /dev/disk/by-uuid 目录中找到任何块设备的 UUID。

磁盘只有在分区后，才会产生 UUID。

➤ /dev/disk/by-uuid

```
[root@oe1db1 ~]# cd /dev/disk
[root@oe1db1 disk]# ls
by-id by-label by-path by-uuid
[root@oe1db1 disk]# ls -la by-uuid/
lrwxrwxrwx 1 root root 10 Dec 10 17:02 e2d11801-784b-40bb-8f96-22a54f12300b
-> ../../sda1
```

➤ blkid /dev/sda1

```
[root@Linux6 proc]# blkid /dev/sda1
/dev/sda1: UUID="d82e08e9-fc04-4bc0-b4bf-fa461c44f1a9" TYPE="ext4"
```

➤ /dev/.udev/db/

```
[oracle@rac11g1 disk]$ cat /dev/.udev/db/*sdf1 | grep -i uuid
S:disk/by-uuid/911cd99b-aabf-4287-b149-77ca3ad0f0a4
E:ID_FS_UUID=911cd99b-aabf-4287-b149-77ca3ad0f0a4
```

➤ **udevadm (Linux6)**

```
[root@Linux6 proc]# udevadm info -q all -n /dev/sdb1 | grep uuid
S: disk/by-uuid/6e8aedde-f808-4e17-a53b-febe81f8d189
```

```
$ udevadm info -a -p $(udevadm info -q path -n /dev/sda)
$ udevadm info -a -p /sys/class/net/eth0
$ udevadm info -q path -n /dev/sda1
$ udevadm control --reload-rules
```

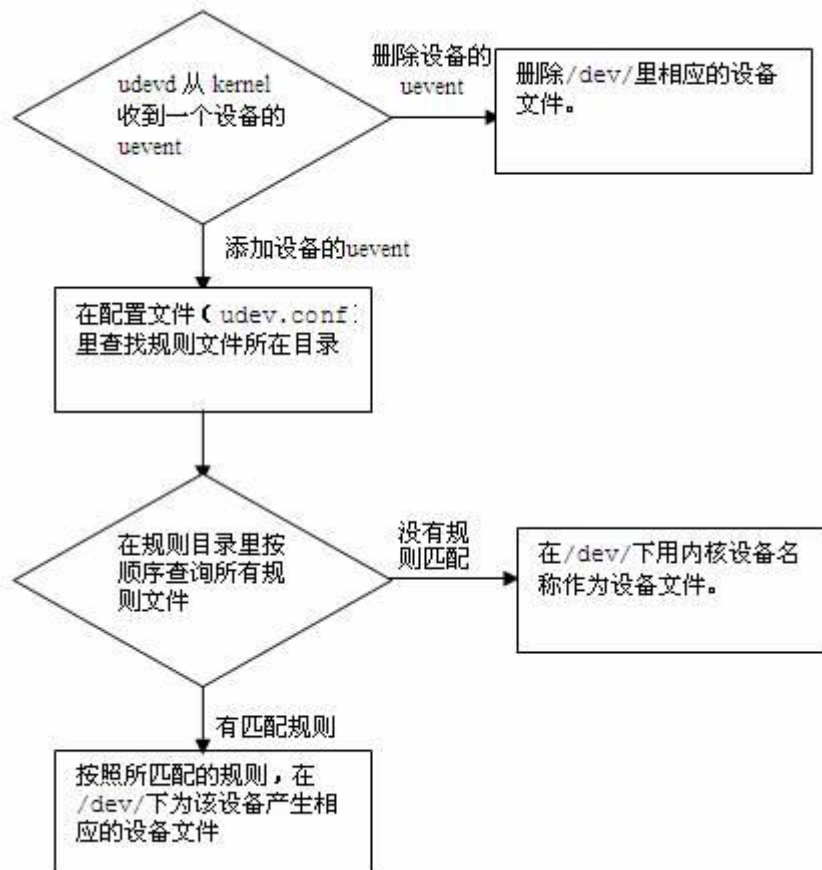
➤ **udevadm (Linux5)**

```
[oracle@rac11g1 disk]$ udevinfo -q all -n /dev/sdf1 | grep -i uuid
S: disk/by-uuid/911cd99b-aabf-4287-b149-77ca3ad0f0a4
E: ID_FS_UUID=911cd99b-aabf-4287-b149-77ca3ad0f0a4
```

```
$ udevinfo -a -p $(udevinfo -q path -n /dev/sda)
$ udevinfo -a -p /sys/class/net/eth0
$ udevinfo -q path -n /dev/sda1
$ udevadm control --reload-rules
```

2.3. 使用 UDEV 进行设备绑定

2.3.1. udev 工作流程图



规则文件是 udev 里最重要的部分，默认是存放在 `/etc/udev/rules.d/` 下。所有的规则文件必须以 `“.rules”` 为后缀名。RHEL 有默认的规则文件，这些默认规则文件不仅为设备产生内核设备名称，还会产生标识性强的符号链接。

2.3.2. 磁盘绑定块设备

```
[root@rhel5 ~]# cd /etc/udev/rules.d/
```

```
[root@rhel5 rules.d]# vi 99-oracle-asmdevices.rules
```

```
KERNEL=="sd*", BUS=="scsi", PROGRAM=="/sbin/scsi_id -g -u -s %p",
RESULT=="14f504e46494c45523559716250332d526e646c2d574e4778", NAME="ASM_DISK1",
OWNER="oracle", GROUP="dba", MODE="0640"

KERNEL=="sd*", BUS=="scsi", PROGRAM=="/sbin/scsi_id -g -u -s %p",
RESULT=="14f504e46494c45524971436245742d486534722d4e61476f", NAME="ASM_DISK2",
OWNER="oracle", GROUP="dba", MODE="0660"
```

```
[root@rhel5 rules.d]# start_udev
```

```
Starting udev: [ OK ]
```

```
[root@rhel5 rules.d]# ls -la /dev/ASM*
brw-r----- 1 oracle dba 8, 16 Jan 10 03:24 /dev/ASM_DISK1
brw-rw---- 1 oracle dba 8, 48 Jan 10 03:24 /dev/ASM_DISK2
```

2.3.3. 磁盘绑定裸设备

```
[root@rhel5 rules.d]# vi 60-raw.rules
```

```
ACTION=="add", KERNEL=="sd*", PROGRAM=="/sbin/scsi_id -g -u -s %p",
RESULT=="14f504e46494c45523559716250332d526e646c2d574e4778", RUN+="/bin/raw
/dev/raw/raw1 %N"
ACTION=="add", KERNEL=="sd*", PROGRAM=="/sbin/scsi_id -g -u -s %p",
RESULT=="14f504e46494c45524971436245742d486534722d4e61476f", RUN+="/bin/raw
/dev/raw/raw2 %N"
KERNEL=="raw1", OWNER="root", GROUP="dba", MODE="640"
KERNEL=="raw2", OWNER="oracle", GROUP="dba", MODE="640"
```

```
[root@rhel5 rules.d]# start_udev
```

```
Starting udev: [ OK ]
```

```
[root@rhel5 ~]# raw -qa
/dev/raw/raw1: bound to major 253, minor 3
/dev/raw/raw2: bound to major 253, minor 2
```

```
[root@rhel5 rules.d]# ls -la /dev/raw/raw*
crw-r----- 1 root dba 162, 1 Jan 10 03:40 /dev/raw/raw1
crw-r----- 1 oracle dba 162, 2 Jan 10 03:40 /dev/raw/raw2
```

2.3.4. 磁盘分区绑定块设备 (RHEL5)

```
# /sbin/scsi_id -g -u -s /block/sdb
KERNEL=="sd?1", BUS=="scsi", PROGRAM=="/sbin/scsi_id -g -u -s /block/$parent",
RESULT=="SATA_VBOX_HARDDISK_VBd306dbe0-df3367e3_", NAME="asm-disk1",
OWNER="oracle", GROUP="dba", MODE="0660"
```

```
[root@rac11g1 ~]# ls -la /dev/sdb*
brw-r----- 1 root disk 8, 16 Jan 16 16:50 /dev/sdb
brw-r----- 1 root disk 8, 17 Jan 16 16:50 /dev/sdb1
brw-r----- 1 root disk 8, 18 Jan 16 16:50 /dev/sdb2
brw-r----- 1 root disk 8, 19 Jan 16 16:50 /dev/sdb3
```

```
[root@rac11g1 ~]# /sbin/scsi_id -g -u -s /block/sdb
```

[SATA_VBOX_HARDDISK_VBe7efc895-388c9867_](#)

分区没有单独的 `scsi_id` 信息，因此同一个磁盘，对应的 `scsi_id` 是一致的。

绑定设备本身：

```
for i in b;
do
    echo "KERNEL=="sd*"\", BUS=="scsi\"", PROGRAM==\"/sbin/scsi_id -g -u -s %p\",
RESULT=="`scsi_id -g -u -s /block/sd$i`\", NAME=\"asm-disk$i\", OWNER=\"grid\",
GROUP=\"asmadmin\", MODE=\"0660\"
done
```

绑定父设备，通常用于磁盘分区方式：

```
for i in f;
do
    echo "KERNEL=="sd?1\", BUS=="scsi\"", PROGRAM==\"/sbin/scsi_id -g -u -s
/block/\$parent\", RESULT=="`scsi_id -g -u -s /block/sd$i`\",
NAME=\"asm-disk$i\", OWNER=\"grid\", GROUP=\"asmadmin\", MODE=\"0660\"
done
```

```
[root@rac11g1 ~]# cd /etc/udev/rules.d/
```

```
[root@rac11g1 rules.d]# vi 99-oracle-asmdevices.rules
```

```
KERNEL=="sd?1", BUS=="scsi", PROGRAM==\"/sbin/scsi_id -g -u -s /block/\$parent\",
RESULT=="SATA\_VBOX\_HARDDISK\_VBe7efc895-388c9867\_", NAME="ASM-OCR",
OWNER="grid", GROUP="asmadmin", MODE="0660"

KERNEL=="sd?2", BUS=="scsi", PROGRAM==\"/sbin/scsi_id -g -u -s /block/\$parent\",
RESULT=="SATA\_VBOX\_HARDDISK\_VBe7efc895-388c9867\_", NAME="ASM-DATA",
OWNER="grid", GROUP="asmadmin", MODE="0660"

KERNEL=="sd?3", BUS=="scsi", PROGRAM==\"/sbin/scsi_id -g -u -s /block/\$parent\",
RESULT=="SATA\_VBOX\_HARDDISK\_VBe7efc895-388c9867\_", NAME="ASM-FRA",
OWNER="grid", GROUP="asmadmin", MODE="0660"
```

注意：使用的是 `/block/\$parent`

```
[root@rac11g1 rules.d]# udevtest /block/sdb/sdb1
```

```
[root@rac11g1 rules.d]# /sbin/udevcontrol reload_rules
```

```
[root@rac11g1 rules.d]# start_udev
```

```
Starting udev: [ OK ]
```

```
[root@rac11g1 rules.d]# ls -la /dev/ASM*
```

```
brw-rw---- 1 grid asmadmin 8, 18 Jan 16 16:51 /dev/ASM-DATA
```

```
brw-rw---- 1 grid asmadmin 8, 19 Jan 16 16:51 /dev/ASM-FRA
```

```
brw-rw---- 1 grid asmadmin 8, 17 Jan 16 16:51 /dev/ASM-OCR
```

2.3.5. 磁盘分区绑定块设备 (RHEL6)

```
# /sbin/scsi_id -g -u -d /dev/sdb
KERNEL=="sd?1", BUS=="scsi", PROGRAM=="/sbin/scsi_id -g -u -d /dev/$parent",
RESULT=="SATA_VBOX_HARDDISK_VBd306dbe0-df3367e3_", NAME="asm-disk1",
OWNER="oracle", GROUP="dba", MODE="0660"
```

进行磁盘绑定前，建议对磁盘进行分区操作，否则容易出现查询不到磁盘设备的问题。

```
for i in b;
do
    echo "KERNEL=="sd*\"", BUS=="scsi\"", PROGRAM==\"/sbin/scsi_id
--whitelisted --replace-whitespace --device=/dev/\$name\"",
RESULT==\"`/sbin/scsi_id --whitelisted --replace-whitespace
--device=/dev/sd$i`\"", NAME=\"asm-disk$i\"", OWNER=\"grid\"", GROUP=\"asmadmin\"",
MODE=\"0660\"""
done
```

```
[root@rhel6 ~]# cd /etc/udev/rules.d/
```

```
[root@rhel6 rules.d]# vi 99-oracle-asmdevices.rules
```

```
KERNEL=="sd*1", BUS=="scsi", PROGRAM=="/sbin/scsi_id --whitelisted
--replace-whitespace --device=/dev/$parent",
RESULT=="1ATA_VBOX_HARDDISK_VBa7a7021e-52c4f243", NAME="ASM-GRID",
OWNER="grid", GROUP="asmadmin", MODE="0660"
```

注意：使用 `/dev/$parent`，也就是父设备的名称。

```
[root@rhel6 rules.d]# udevadm test /sys/block/sdb/sdb1
```

```
[root@rhel6 rules.d]# udevadm info -q path -n /dev/sdb
```

```
/devices/pci0000:00/0000:00:0d.0/host3/target3:0:0/3:0:0:0/block/sdb
```

```
[root@rhel6 ~]# udevadm info -a -p /sys/block/sdb
```

```
[root@rhel6 rules.d]# start_udev
```

```
Starting udev: [ OK ]
```

```
[root@rhel6 rules.d]# ls -la /dev/ASM*
```

```
brw-rw---- 1 grid asmadmin 8, 17 Jan 17 15:15 /dev/ASM-GRID
```

2.3.6. 磁盘分区绑定块设备 (RHEL7)

```
# /usr/lib/udev/scsi_id -g -u -d /dev/sdb
```

```
KERNEL=="sd?1", SUBSYSTEM=="block", PROGRAM=="/usr/lib/udev/scsi_id -g -u -d /dev/$parent", RESULT=="SATA_VBOX_HARDDISK_VBd306dbe0-df3367e3_", SYMLINK+="asm-disk1", OWNER="oracle", GROUP="dba", MODE="0660"
```

```
for i in b c d e;
do
    echo "KERNEL=="sd?1\", SUBSYSTEM=="block\",
PROGRAM==\"/usr/lib/udev/scsi_id -g -u -d /dev/\$parent\",
RESULT==\"`/usr/lib/udev/scsi_id -g -u -d /dev/sd$i`\",
SYMLINK+=\"oracleasm/asm-disk$i\", OWNER=\"grid\", GROUP=\"asmadmin\",
MODE=\"0660\"
done
```

```
[root@rac12c1 ~]# vi /etc/udev/rules.d/98-oracle-asmdevices.rules
KERNEL=="sd?1", SUBSYSTEM=="block", PROGRAM=="/usr/lib/udev/scsi_id -g -u -d /dev/$parent", RESULT=="1ATA_VBOX_HARDDISK_VBb8bf6c10-a75ce301", SYMLINK+="oracleasm/asm-diskb", OWNER="grid", GROUP="asmadmin", MODE="0660"
KERNEL=="sd?1", SUBSYSTEM=="block", PROGRAM=="/usr/lib/udev/scsi_id -g -u -d /dev/$parent", RESULT=="1ATA_VBOX_HARDDISK_VB9ad089cd-b6afbc7b", SYMLINK+="oracleasm/asm-diskc", OWNER="grid", GROUP="asmadmin", MODE="0660"
KERNEL=="sd?1", SUBSYSTEM=="block", PROGRAM=="/usr/lib/udev/scsi_id -g -u -d /dev/$parent", RESULT=="1ATA_VBOX_HARDDISK_VB0eb82c64-8cd03011", SYMLINK+="oracleasm/asm-diskd", OWNER="grid", GROUP="asmadmin", MODE="0660"
KERNEL=="sd?1", SUBSYSTEM=="block", PROGRAM=="/usr/lib/udev/scsi_id -g -u -d /dev/$parent", RESULT=="1ATA_VBOX_HARDDISK_VB7ad79a8f-8597796e", SYMLINK+="oracleasm/asm-diske", OWNER="grid", GROUP="asmadmin", MODE="0660"
```

```
[root@rac12c1 ~]# /sbin/udevadm control --reload-rules
```

```
[root@rac12c1 ~]# ls -la /dev/oracleasm/*
```

```
lrwxrwxrwx. 1 root root 7 May 12 15:50 /dev/oracleasm/asm-diskb -> ../sdb1
lrwxrwxrwx. 1 root root 7 May 12 15:50 /dev/oracleasm/asm-diskc -> ../sdc1
lrwxrwxrwx. 1 root root 7 May 12 15:51 /dev/oracleasm/asm-diskd -> ../sdd1
lrwxrwxrwx. 1 root root 7 May 12 15:51 /dev/oracleasm/asm-diske -> ../sde1
```

RHEL5/RHEL6/RHEL7 的 scsi_id 命令，有较大的区别。

RHEL5

```
# /sbin/scsi_id -g -u -s /block/sda
```

RHEL6

```
# /sbin/scsi_id --whitelisted --replace-whitespace --device=/dev/sda
```

```
# /sbin/scsi_id -g -u -d /dev/sda
```

RHEL7

```
# /usr/lib/udev/scsi_id -g -u -d /dev/sdb
```

参考文档:

<https://oracle-base.com/articles/linux/udev-scsi-rules-configuration-in-oracle-linux>

2.3.7. 磁盘分区绑定裸设备

```
[root@rac11g1 ~]# ls -la /dev/sdb*
brw-r----- 1 root disk 8, 16 Jan 16 16:50 /dev/sdb
brw-r----- 1 root disk 8, 17 Jan 16 16:50 /dev/sdb1
brw-r----- 1 root disk 8, 18 Jan 16 16:50 /dev/sdb2
brw-r----- 1 root disk 8, 19 Jan 16 16:50 /dev/sdb3
```

```
[root@rac11g1 ~]# /sbin/scsi_id -g -u -s /block/sdb
SATA_VBOX_HARDDISK_VBe7efc895-388c9867_
```

分区没有单独的 `scsi_id` 信息, 因此同一个磁盘, 对应的 `scsi_id` 是一致的。

```
[root@rac11g1 ~]# cd /etc/udev/rules.d/
```

```
[root@rac11g1 rules.d]# vi 60-raw.rules
```

```
ACTION=="add", KERNEL=="sd?1", PROGRAM=="/sbin/scsi_id -g -u -s /block/$parent",
RESULT=="SATA_VBOX_HARDDISK_VBe7efc895-388c9867_", RUN+="/bin/raw
/dev/raw/raw1 %N"
ACTION=="add", KERNEL=="sd?2", PROGRAM=="/sbin/scsi_id -g -u -s /block/$parent",
RESULT=="SATA_VBOX_HARDDISK_VBe7efc895-388c9867_", RUN+="/bin/raw
/dev/raw/raw2 %N"
KERNEL=="raw1", OWNER="root", GROUP="dba", MODE="640"
KERNEL=="raw2", OWNER="oracle", GROUP="dba", MODE="640"
```

```
[root@rac11g1 rules.d]# ls -la /dev/raw/raw*
crw-r----- 1 root dba 162, 1 Jan 16 17:13 /dev/raw/raw1
crw-r----- 1 oracle dba 162, 2 Jan 16 17:13 /dev/raw/raw2
```

2.3.8. 多路径 DM 绑定

```
KERNEL=="dm-*", ENV{DM_UUID}=="part1-mpath-360080e500041b060000007715523016e",
NAME="asmdisk-ocrvdisk1", OWNER="grid", GROUP="asmadmin", MODE="0660"
KERNEL=="dm-*", ENV{DM_UUID}=="part1-mpath-360080e500041b0600000076b552300f2",
NAME="asmdisk-ocrvdisk2", OWNER="grid", GROUP="asmadmin", MODE="0660"
KERNEL=="dm-*", ENV{DM_UUID}=="part1-mpath-360080e500041b0600000076f55230161",
NAME="asmdisk-ocrvdisk3", OWNER="grid", GROUP="asmadmin", MODE="0660"
```



```

KERNEL=="dm-*", ENV{DM_UUID}=="part1-mpath-360080e500041b06000000777552301a5",
NAME="asmdisk-mrcf1", OWNER="grid", GROUP="asmadmin", MODE="0660"
KERNEL=="dm-*", ENV{DM_UUID}=="part1-mpath-360080e500041b0600000077755230197",
NAME="asmdisk-mrcf2", OWNER="grid", GROUP="asmadmin", MODE="0660"
KERNEL=="dm-*", ENV{DM_UUID}=="part1-mpath-360080e500041b0600000077755230187",
NAME="asmdisk-arch2", OWNER="grid", GROUP="asmadmin", MODE="0660"
KERNEL=="dm-*", ENV{DM_UUID}=="part1-mpath-360080e500041d1980000071d552302f6",
NAME="asmdisk-data1", OWNER="grid", GROUP="asmadmin", MODE="0660"
KERNEL=="dm-*", ENV{DM_UUID}=="part1-mpath-360080e500041b060000007775523020e",
NAME="asmdisk-data2", OWNER="grid", GROUP="asmadmin", MODE="0660"
KERNEL=="dm-*", ENV{DM_UUID}=="part1-mpath-360080e500041d1980000071f55230313",
NAME="asmdisk-data3", OWNER="grid", GROUP="asmadmin", MODE="0660"
KERNEL=="dm-*", ENV{DM_UUID}=="part1-mpath-360080e500041d1980000072155230331",
NAME="asmdisk-data4", OWNER="grid", GROUP="asmadmin", MODE="0660"
KERNEL=="dm-*", ENV{DM_UUID}=="part1-mpath-360080e500041b0600000077b5523022b",
NAME="asmdisk-data5", OWNER="grid", GROUP="asmadmin", MODE="0660"

```

2.3.9. UDEV 的值和可调用的替换操作符

\$kernel, %k: 设备的内核设备名称, 例如: sda、cdrom。

\$number, %n: 设备的内核号码, 例如: sda3 的内核号码是 3。

\$devpath, %p: 设备的 devpath 路径。

\$id, %b: 设备在 devpath 里的 ID 号。

\$sysfs{file}, %s{file}: 设备的 sysfs 里 file 的内容。其实就是设备的属性值。

例如: **\$sysfs{size}** 表示该设备 (磁盘) 的大小。

\$env{key}, %E{key}: 一个环境变量的值。

\$major, %M: 设备的 major 号。

\$minor %m: 设备的 minor 号。

\$result, %c: PROGRAM 返回的结果。

\$parent, %P: 父设备的设备文件名。

\$root, %r: udev_root 的值, 默认是 /dev/。

\$tempnode, %N: 临时设备名。

%%: 符号 % 本身。

\$\$: 符号 \$ 本身。

2.3.10. UDEV 常用命令

2.3.10.1 测试 udev 设备配置信息 udevadm (RHEL 6)

```
[oracle@rhel6 ~]$ udevadm test /block/sdb/sdb1
```

2.3.10.2 测试 udev 设备配置信息 udevtest (RHEL 5)

udevtest 会针对一个设备，在不需要 **uevent** 触发的情况下模拟一次 **udev** 的运行，并输出查询规则文件的过程、所执行的行为、规则文件的执行结果。通常使用 **udevtest** 来调试规则文件。

```
[root@rac11g1 rules.d]# udevtest /block/sdb/sdb1
udev_node_add: creating device node '/dev/ASM-OCR', major = '8', minor = '17',
mode = '0660', uid = '1001', gid = '1010'
udev_node_add: creating symlink
'/dev/disk/by-id/scsi-SATA_VBOX_HARDDISK_VBe7efc895-388c9867-part1' to
'../../ASM-OCR'
udev_node_add: creating symlink
'/dev/disk/by-path/pci-0000:00:0d.0-scsi-1:0:0:0-part1' to ' ../../ASM-OCR'
```

2.3.10.3 重新加载 UDEV 规则、启动 UDEV 服务

```
# /sbin/udevadm control --reload-rules
# /sbin/start_udev
```

2.3.10.4 udevinfo

```
[oracle@ednmr1p1 db]$ udevinfo -a -p /sys/block/sdf/sdf1
```

```
looking at device '/block/sdf/sdf1':
  KERNEL=="sdf1"
  SUBSYSTEM=="block"
  SYSFS{stat}=="      "
  SYSFS{size}=="1044162"
  SYSFS{start}=="63"
  SYSFS{dev}=="8:81"
```

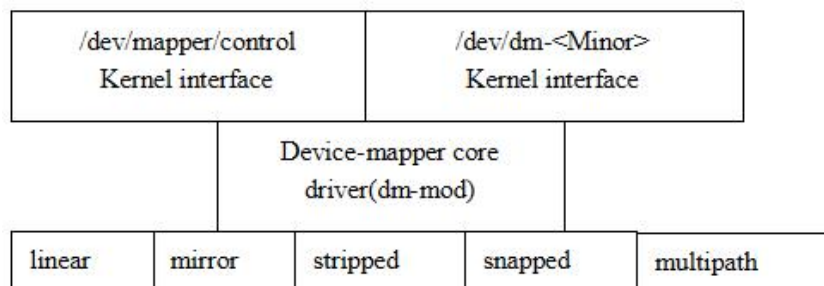
```
[root@ednmr1p1 ASM]# scsi_id -g -x -s /block/sdf/sdf1
ID_VENDOR=ATA
ID_MODEL=VBOX_HARDDISK
ID_REVISION=1.0
ID_SERIAL=SATA_VBOX_HARDDISK_VBb297a092-07fadff5
ID_TYPE=disk
ID_BUS=scsi
```

2.4. 管理设备的多路径 I/O

多路径是服务器与跨多个物理路径（这些路径在服务器中的主机总线适配器和设备储存控制器之间）的同一物理或逻辑块储存设备通讯的能力，通常是在光纤通道（FC）或 iSCSI SAN 环境中。

Linux 的多路径提供连接容错，并可以跨多个活动连接提供负载平衡。当多路径已配置并且正在运行时，它会自动隔离和识别设备连接故障，并重路由 I/O 以改变连接。

2.4.1. Device-mapper



➤ mapped device

Mapped device is a logical device provided by device-mapper driver.

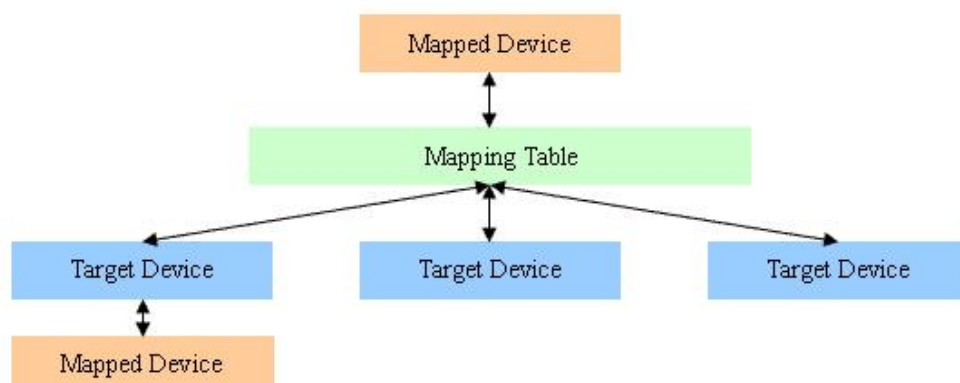
➤ mapping table

Mapping table represents a mapping from a mapped device to target devices.

➤ target device

Target device is a modularized plugin. The device-mapper filters and redirects I/O requests with it. In figure 1 we can see several kinds of typical target driver:

1. linear for LVM2
2. mirror for RAID
3. stripped for LVM2
4. snapshot for LVM2
5. multipath for dm-multipath



Understanding Device-mapper in Linux 2.6 Kernel (Doc ID 456239.1)

2.4.2. 规划多路径

2.4.2.1 多路径准则

- 先决条件

1. 多路径在设备级别进行管理。
2. 用于多路径设备的储存阵列必须支持多路径。
3. 仅当在服务器中的主机总线适配器和块储存设备的主机总线控制器之间存在多个物理路径时,才需要配置多路径。

- 高可用性解决方案

群集储存资源的高可用性解决方案基于每个节点上的多路径服务运行。请确保每个节点上的 `/etc/multipath.conf` 文件中的配置设置在整个集群中保持一致。

请确保所有设备上的多路径设备名称相同,可采用的方法如下:

1. 使用 **UUID** 和别名确保多路径设备名称在集群中的所有节点上保持一致。别名在所有节点上必须唯一。从该节点将 `/etc/multipath.conf` 文件复制到集群中所有其他节点的 `/etc/` 目录。
2. 当使用多路径映射设备的链接时,请确保在 `/dev/disk/by-id` 目录中指定 `dm-uuid*` 名称或别名,而不是设备的固定路径实例。
3. 将 `user_friendly_names` 配置选项设置为“no”以禁用它。用户友好名称对某个节点是唯一的,但集群中各个节点上的设备不得指派相同的用户友好名称。

- 卷管理器

诸如 `L VM2` 以及集群的 `L VM2` 等卷管理器在多路径的基础上运行。必须先配置设备的多路径,才能使用 `L VM2` 或 `CL VM2` 在该设备上创建段管理器和文件系统。

2.4.2.2 为多路径设备使用 WWID、用户友好名称以及别名

多路径设备可以按 **WWID**、用户友好名称或您指派的别名进行唯一标识。格式为 `/dev/sdn` 和 `/dev/dm-n` 的设备节点名称在重引导时会发生更改,且每次都可能会被指派给不同的设备。而设备的 **WWID**、用户友好名称以及别名则在重引导期间始终不变,因此是标识设备的首选方式。

在 `/etc/multipath.conf` 文件中定义设备别名时, 请**确保使用每个设备的 WWID** (如 `3600508e000000009e6baa6f609e7908`) 而不是它的 **WWN** (以 `0x` 替换设备 ID 的第一个字符, 如 `0x600508e0000000009e6baa6f609e7908`)。

重要

当使用多路径映射设备的链接时, 请确保在 `/dev/disk/by-id` 目录中指定 `dm-uuid*` 名称或别名, 而不是设备的固定路径实例。

The `/dev/dm-N` devices are used internally by device-mapper-multipath and are non-persistent across reboot, so should not be used. The `/dev/mpath/` devices are created for multipath devices to be visible together, however, may not be available during early stages of boot, so, again, should not be used. However, `/dev/mapper/` devices are persistent, created sufficiently early during the boot process and makes use of their defined aliases - use only these devices to access and interact with multipathed devices. 建议使用 `/dev/mapper` 下的设备文件。

2.4.2.3 在多路径设备上使用 LVM2

请确保 `lvm.conf` 的配置文件指向多路径设备名称而不是固定路径名称。

将多路径设备过滤器添加到 `/etc/lvm/lvm.conf` 文件

默认情况下, LVM2 不能识别多路径设备。要让 LVM2 将多路径设备识别为可能的物理卷, 则必须修改 `/etc/lvm/lvm.conf`, 以通过多路径 I/O 层扫描多路径设备。

通过添加多路径过滤器, 可以防止 LVM 扫描和使用代表 SAN (`/dev/sd*`) 的单个路径的原始设备节点的物理路径。请确保指定过滤器路径, 以便 LVM 在配置多路径后仅会扫描设备 (`/dev/disk/by-id/dm-uuid-.*-mpath-.*`) 的设备映射器名称。

1. 生成 lvm 配置文件

```
[root@openfiler ~]# lvm dumpconfig > /etc/lvm/lvm.conf
```

2. 将 `/etc/lvm/lvm.conf` 中的 `filter` 和 `types` 条目作如下更改:

```
filter = [ "a|/dev/disk/by-id/.*|", "r|.*|" ]
types = [ "device-mapper", 1 ]
```

这使 LVM2 仅扫描 `by-id` 路径而拒绝其他。

3. 如果还要在非多路径设备上使用 LVM2, 请在 `filter` 和 `types` 条目中作必要的调整以符合您的设置。否则, 针对多路径修改 `lvm.conf` 文件后, 不能使用 `pvscan` 看到其他 LVM 设备。

```
filter = [ "a|/dev/sda.*|", "a|/dev/disk/by-id/.*|", "r|.*|" ]
types = [ "device-mapper", 253 ]
```

使用 `dmsetup` 命令检查每一个多路径设备是否拥有预期数量的路径：

```
[root@nmacct2 ~]# dmsetup ls --tree
system_vg-redo1_3 (252:16)
└─ (8:32)
system_vg-sysaux (252:7)
└─ (8:32)
system_vg-redo2_3 (252:20)
└─ (8:32)
```

2.4.2.4 对多路径设备使用 mdadm

`mdadm` 工具要求使用 `ID` 而不是设备节点路径访问设备。因此，应按照以下方式设置 `/etc/mdadm.conf` 文件中的 `DEVICE` 项，以确保在配置多路径后仅对设备映射器名称进行扫描：

```
DEVICE /dev/disk/by-id/dm-uuid-.*mpath-.*
```

2.4.3. 多路径管理工具

多路径支持基于 `Linux 2.6` 内核的设备映射程序多路径模块和 `multipath-tools` 用户空间包。可以使用 `Multiple Devices Administration` 实用程序（`MDADM`, `mdadm`）查看多路径设备的状态。

```
[root@nmacct2 ~]# mdadm --version
mdadm - v2.6.9 - 10th March 2009
```

SCSI 设备名的格式为 `/dev/sdN`，其中 `N` 是为该设备自动生成的字母，从 `a` 开始，并在设备创建时顺序发放，例如 `/dev/sda`、`/dev/sdb` 等。如果设备数超过 `26`，则会使用两个字母，这样在 `/dev/sdz` 后的下一个设备将命名为 `/dev/sdaa`、`/dev/sdab` 等。

工具	描述
<code>multipath</code>	扫描系统中的多路径设备并组装它们。
<code>multipathd</code>	等候映射事件，然后执行 <code>multipath</code> 。
<code>devmap-name</code>	为设备映射（ <code>devmaps</code> ）的 <code>udev</code> 提供有意义的设备名。
<code>kpartx</code>	将线性 <code>devmaps</code> 映射到多路径设备上的分区，使得能够为设备上的分区创建多路径监视。

2.4.3.1 多路径磁盘格式化

```
# kpartx -a /dev/mapper/mpath#
```

2.4.4. 为多路径配置系统

2.4.4.1 为多路径准备 SAN 设备

注意

确保您正在使用的 HBA 驱动程序没有启用本机多路径。

如果 HBA 驱动程序没有看到这些 LUN，则可以使用 `lsscsi` 检查操作系统是否正确看到这些 SCSI 设备。

2.4.5. 创建或修改 `/etc/multipath.conf` 文件

只有创建 `/etc/multipath.conf`，该文件才会存在。除非您创建多路径配置文件并进行个性化设置，否则在运行守护程序 `multipathd` 时，将自动应用默认的多路径设备设置。

无论您何时创建或修改 `/etc/multipath.conf` 文件，当保存该文件时都不会自动应用更改。

2.4.5.1 创建 `/etc/multipath.conf` 文件

```
cp /usr/share/doc/device-mapper-multipath-0.4.9/multipath.conf  
/etc/multipath.conf
```

2.4.5.2 `/etc/multipath.conf` 文件中的部分

- **defaults**

多路径 I/O 的常规默认设置。

- **blacklist**

列出不作为候选多路径而丢弃的设备名称。可以通过设备节点名称(`devnode`)、设备 WWID (`wwid`) 或设备供应商或产品字符串 (`device`) 来识别设备。

- **blacklist_exceptions**

列出即使包含在黑名单中仍视为候选多路径的设备的设备名称。

- **devices**

指定单个储存控制器的设置。这些值会重写配置文件的 **defaults** 部分中指定的值。

- **multipaths**

指定单个多路径设备的设置。

2.4.5.3 校验 `/etc/multipath.conf` 文件中的多路径设置

无论您何时创建或修改 `/etc/multipath.conf` 文件，当保存该文件时都不会自动应用更改。您可以在更新多路径映射之前对该设置执行“试运行”，以校验多路径设置。

在服务器命令提示符处输入：

```
[root@nmacct2 ~]# multipath -v2 -d
```

在试运行中使用详细级别 `-v3`，即可查看所有检测到的路径、多路径和设备映射。

2.4.5.4 应用 `/etc/multipath.conf` 文件更改以更新多路径映射

对 `/etc/multipath.conf` 文件的更改在 `multipathd` 运行时不会生效。进行更改后，保存并关闭该文件，然后执行以下操作以应用更改并更新多路径映射：

1. 停止 `multipathd` 服务

```
[root@nmacct2 /]# service multipathd stop  
Stopping multipathd daemon: [ OK ]
```

2. 通过输入以下命令，清除旧的多路径绑定

```
/sbin/multipath -F
```

3. 通过输入以下命令，创建新的多路径绑定

```
/sbin/multipath -v2 -l
```

4. 启动 `multipathd` 服务

```
[root@nmacct2 /]# service multipathd start  
Starting multipathd daemon: [ OK ]
```

2.4.6. 将非多路径设备列入黑名单

`/etc/multipath.conf` 文件应包含一个 **blacklist** 部分，列出所有非多路径设备。

```
blacklist {  
    wwid 26353900f02796769
```



```
devnode "^(ram|raw|loop|fd|md|dm-|sr|scd|st|sda)[0-9]*"
devnode "^hd[a-z][0-9]*"
devnode "^cciss!c[0-9]d[0-9].*"
}
```

修改 `/etc/multipath.conf` 文件之后，必须在您的系统上运行 `mkinitrd` 以重新创建 `initrd`，然后重新启动服务器，以使更改生效。

之后，当发出 `multipath -ll` 命令时，这些本地设备应不再列于多路径映射中。

2.4.7. 多路径设备名称类型

名称类型	描述
WWID（默认）	WWID（全球标识符）序列是保证全球唯一且不会更改的多路径设备标识符。在多路径中使用的默认名称是 <code>/dev/disk/by-id</code> 目录中的逻辑单元的 ID。
用户友好	<code>/dev/mapper</code> 目录中的设备映射程序多路径设备名称也参考逻辑单元的 ID。这些多路径设备名称都是用户友好的名称，格式为 <code>/dev/mapper/mpath<n></code> ，比如 <code>/dev/mapper/mpath0</code> 。名称是唯一且永久的，因为它们使用 <code>/var/lib/multipath/bindings</code> 文件跟踪 UUID 和用户友好的名称之间的关联。
别名	别名是由管理员为多路径设备提供的全局唯一名称。别名覆盖 WWID 和用户友好的 <code>/dev/mapper/mpathN</code> 名称。

2.4.8. 在不重引导的情况下扫描新设备

```
[root@openfiler dev]# whereis rescan-scsi-bus
rescan-scsi-bus: /usr/sbin/rescan-scsi-bus.sh
[root@openfiler dev]# /usr/sbin/rescan-scsi-bus.sh
Host adapter 0 (ahci) found.
Host adapter 1 (ahci) found.
Host adapter 2 (ahci) found.
Host adapter 3 (mptspi) found.
```

2.4.9. 查看多路径 I/O 状态

```
[root@openfiler ~]# multipath -ll

3600601607cf30e00184589a37a31d911
[size=127 GB][features="0"][hwhandler="1 emc"]
```

```
\_ round-robin 0 [active][first]
\_ 1:0:1:2 sdav 66:240 [ready ][active]
\_ 0:0:1:2 sdr 65:16 [ready ][active]
\_ round-robin 0 [enabled]
\_ 1:0:0:2 sdag 66:0 [ready ][active]
\_ 0:0:0:2 sdc 8:32 [ready ][active]
```

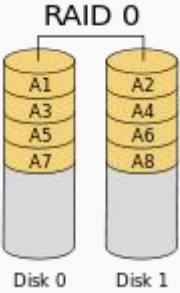
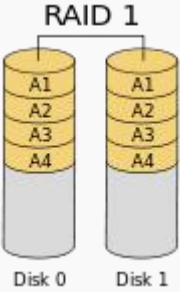
对于每个设备，它显示该设备的 ID、大小、功能和硬件处理程序。

会针对每个路径显示下列信息：

- 1. 物理地址，格式为 `host:bus:target:lun`，例如 `1:0:1:2`
- 2. 设备节点名，例如 `sda`
- 3. 主要:次要编号
- 4. 设备的状态

2.5. 软件 RAID 配置

RAID（独立磁盘冗余阵列）的用途是将多个硬盘分区合并成一个大的虚拟硬盘，以便优化性能和/或数据安全性。大多数 RAID 控制器使用 SCSI 协议，因为对大量硬盘，它可用比 IDE 协议更高效的方式寻址，更适于命令的并行处理。

Level	Minimum # of drives	Figure
RAID 0	2	
RAID 1	2	

RAID 2	3	
RAID 3	3	
RAID 4	3	
RAID 5	3	
RAID 6	4	
RAID 10	4	

重要

在群集文件系统（例如 OCFS2）下不支持软件 RAID，因为 RAID 不支持并行

激活。如果想要 RAID 用于 OCFS2，则需要 RAID 由储存子系统处理。

2.5.1. 了解 RAID 级别

2.5.1.1 RAID 0

此级别通过将每个文件按块分放到多个磁盘驱动器上，提高了数据访问性能。这实际上并不是真正的 RAID，因为它未提供数据备份，

2.5.1.2 RAID 1

此级别为数据提供了充分的安全性，因为它将数据按 1:1 复制到另一块硬盘上。这种方法称为硬盘镜像。

2.5.1.3 RAID 5

RAID 5 是级别 0 和级别 1 在性能和冗余方面经优化后的折衷方案。硬盘空间等于使用的磁盘数减 1。数据分布在这些硬盘上，这一点与 RAID 0 相同。但出于安全原因，在其中一个分区上创建了奇偶校验块。这些块通过 XOR 互相链接，并在系统出现故障时，通过启用相应的校验块重建内容。对于 RAID 5，在同一时间只能有一块硬盘出现故障。如果一块硬盘出现故障，则必须尽快将其更换，以防止丢失数据。

2.5.2. 查错软件 RAID

查看 /proc/mdstats 文件以确定 RAID 分区是否受损。如果系统出现故障，请关闭 Linux 系统并用以同样方式分区的新硬盘替换出现问题的硬盘。

2.5.3. 创建 RAID

```
[root@openfiler ~]# mdadm --create /dev/md0 --level=0 --chunk=256
--raid-devices=2 /dev/sdf /dev/sdg
mdadm: /dev/sdf appears to be part of a raid array:
   level=raid0 devices=0 ctime=Thu Jan 1 08:00:00 1970
mdadm: partition table exists on /dev/sdf but will be lost or
   meaningless after creating array
mdadm: /dev/sdg appears to be part of a raid array:
   level=raid0 devices=0 ctime=Thu Jan 1 08:00:00 1970
mdadm: partition table exists on /dev/sdg but will be lost or
```

```
meaningless after creating array
Continue creating array? y
mdadm: Defaulting to version 1.2 metadata
mdadm: array /dev/md0 started.

[root@openfiler ~]# ls -la /dev/md0
brw-rw---- 1 root disk 9, 0 Dec 12 01:37 /dev/md0
```

2.5.4. 嵌套的 RAID 设备

2.5.4.1 RAID 1+0:

先构建 RAID 1（镜像）阵列，然后组合形成 RAID 0（条带化）阵列。

RAID 1+0 提供高级别的 I/O 性能、数据冗余、和磁盘容错。因为 RAID 0 中的每个成员设备都分别镜像，因此可以容忍多个磁盘故障，并且只要失败的磁盘在不同的镜像中，数据仍然可用。

2.5.4.2 RAID 0+1:

先构建 RAID 0（条带化）阵列，然后组合形成 RAID 1（镜像）阵列。

RAID 0+1 提供高级别的 I/O 性能和数据冗余，但 1+0 的容错稍差。如果在镜像的一端多个磁盘失败，则另一个镜像可用。但是，如果在镜像的两端同时丢失磁盘，则所有数据会丢失。

2.5.4.3 创建 RAID 10

嵌套的 RAID 1+0 的构建方法是，创建两个或更多 RAID 1（镜像），然后使用它们作为 RAID 0 中的组件设备。

1. 创建 2 个软件 RAID 1 设备，为每个 RAID 1 设备使用两个不同的设备。

```
mdadm --create /dev/md0 --run --level=1 --raid-devices=2 /dev/sdb1 /dev/sdc1
mdadm --create /dev/md1 --run --level=1 --raid-devices=2 /dev/sdd1 /dev/sde1
```

2. 创建嵌套的 RAID 1+0 设备。

```
mdadm --create /dev/md2 --run --level=0 --chunk=64 --raid-devices=2 /dev/md0
/dev/md1
```

3. 在 RAID 1+0 设备 /dev/md2 上创建一个文件系统

```
mkfs.reiserfs /dev/md2
```

2.6. 经由 IP 网络的大容量储存：iSCSI

iSCSI 是一种储存联网协议，便于在块储存设备和服务器之间通过 TCP/IP 网络将数据作为 SCSI 包传送。iSCSI 目标软件在目标服务器上运行，并将逻辑单元定义为 iSCSI 目标设备。iSCSI 发起程序软件在不同服务器上运行并连接到目标设备，以使此服务器上的储存设备可用。

不支持在生产环境中的同一服务器上运行 iSCSI 目标软件和 iSCSI 发起程序软件。

iSCSI 目标和 iSCSI 发起程序服务器通过在 LAN 中发送 IP 级的 SCSI 包进行通讯。

2.7. 以太网光纤通道储存：FCoE

开放以太网光纤通道（FCoE）发起程序软件允许配有以太网适配器的服务器经由以太网网络连接光纤通道储存子系统。

2.8. 磁盘管理常用命令

2.8.1. blkid

blkid - command-line utility to locate/print block device attributes

```
[root@rhel6 by-uuid]# blkid /dev/sda1
```

```
/dev/sda1: UUID="18b7e913-5308-46ba-a317-b1da7fd4d340" TYPE="ext4"
```

```
[root@rhel6 ~]# cd /dev/disk/by-uuid/
```

```
[root@rhel6 by-uuid]# ls -la * | awk '{print $9, $10, $11}'
```

```
18b7e913-5308-46ba-a317-b1da7fd4d340 -> ../../sda1
```

```
5226c539-7005-47a9-bcda-26a0e6f800a6 -> ../../dm-0
```

```
67b8ebb7-ac34-46b2-89c1-b4e1965c83f1 -> ../../dm-1
```

```
[root@rhel6 by-uuid]# more /etc/fstab | grep UUID
```

```
UUID=18b7e913-5308-46ba-a317-b1da7fd4d340    /boot    ext4    defaults
1 2
```

```
[root@rhel6 by-uuid]# df -h | grep sda1
```

```
/dev/sda1    485M    37M    423M    9% /boot
```

2.8.2. /etc/fstab

```
[root@rhel6 ~]# more /etc/fstab
```

/dev/mapper/vg_rhel6-lv_root	/	ext4	defaults	1	1
/dev/sda1	/boot	ext4	defaults	1	2
/dev/mapper/vg_rhel6-lv_swap	swap	swap	defaults	0	0
tmpfs	/dev/shm	tmpfs	defaults	0	0
devpts	/dev/pts	devpts	gid=5,mode=620	0	0
sysfs	/sys	sysfs	defaults	0	0
proc	/proc	proc	defaults	0	0

一行有 6 个字段。因为字段是定位的，所有值必须被指定。

第一列：文件系统

这可能是一个设备名，比如 `/dev/sda1` 或者一个标签（`LABEL=`），或者是 `UUID`（`UUID=`）。

第二列：挂载点

第三列：类型

指定文件系统类型。`CD/DVD` 驱动器通常只支持 `ISO9660` 和 `UDF` 文件系统其中一个，因此您可能要在一个逗号分隔的列表中指定多种可能性。

第四列：选项

指定安装选项，如果您想用默认安装选项，指定 `defaults`。您需要了解的选项有：

參數	內容意義
async/sync 非同步/同步	設定磁碟是否以非同步方式運作！預設為 async (效能較佳)
auto/noauto 自動/非自動	當下達 <code>mount -a</code> 時，此檔案系統是否會被主動測試掛載。預設為 auto 。
rw/ro 可讀寫/唯讀	讓該分割槽以可讀寫或者是唯讀的型態掛載上來，如果你想要分享的資料是不給使用者隨意變更的，這裡也能夠設定為唯讀。則不論在此檔案系統的檔案是否設定 <code>w</code> 權限，都無法寫入喔！
exec/noexec 可執行/不可執行	限制在此檔案系統內是否可以進行『執行』的工作？如果是純粹用來儲存資料的，那麼可以設定為 noexec 會比較安全，相對的，會比較麻煩！
user/nouser 允許/不允許使用者掛載	是否允許使用者使用 <code>mount</code> 指令來掛載呢？一般而言，我們當然不希望一般身份的 user 能使用 <code>mount</code> 囉，因為太不安全了，因

	此這裡應該要設定為 nouser 囉！
suid/nosuid 具有/不具有 suid 權限	該檔案系統是否允許 SUID 的存在？如果不是執行檔放置目錄，也可以設定為 nosuid 來取消這個功能！
usrquota	注意名稱是『 usrquota 』不要拼錯了！這個是在啟動 filesystem 支援磁碟配額模式，更多資料我們在第四篇再談。
grpquota	注意名稱是『 grpquota 』，啟動 filesystem 對群組磁碟配額模式的支援。
defaults	同時具有 rw, suid, dev, exec, auto, nouser, async 等參數。基本上，預設情況使用 defaults 設定即可！

第五列：dump

指定 **dump** 命令是否应考虑 **ext2** 或 **ext3** 文件系统作 **dump** 备份。值为 **0** 则通知 **dump** 忽略该文件系统。

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** 备份动作，通常这个数值为 **0** 或 **1**！

第六列：fsck

开机的过程中，系统预设会以 **fsck** 检验我们的 **filesystem** 是否完整 (**clean**)。

日常添加文件系统时，第五列和第六列指定为 **0** 即可。

2.9. 磁盘管理常用文件

2.9.1. /proc/partitions

```
[root@Linux6 ~]# cat /proc/partitions
major minor #blocks name
```

```
11      0    1048575 sr0
 8      0    10485760 sda
 8      1      204800 sda1
 8      2    2097152 sda2
 8      3    8182784 sda3
 8     16    20971520 sdb
 8     17    20964793 sdb1
```



```
[root@Linux6 ~]# for i in `cat /proc/partitions | awk '{print $4}' |grep sd |  
grep [a-z]$`; do echo "### $i: `/lib/udev/scsi_id --whitelisted  
--device=/dev/$i`"; done
```

```
### sda: 1ATA      VBOX HARDDISK          VB70b6953a-cc277e5f  
### sdb: 1ATA      VBOX HARDDISK          VBdca91247-5235e07b
```

3. LVM 配置

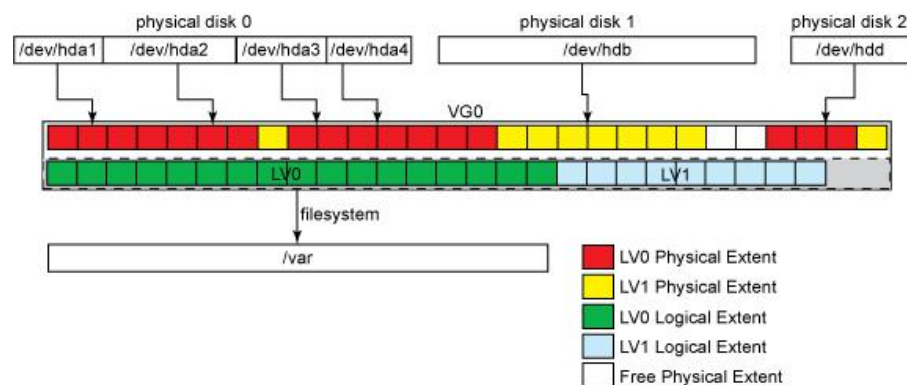
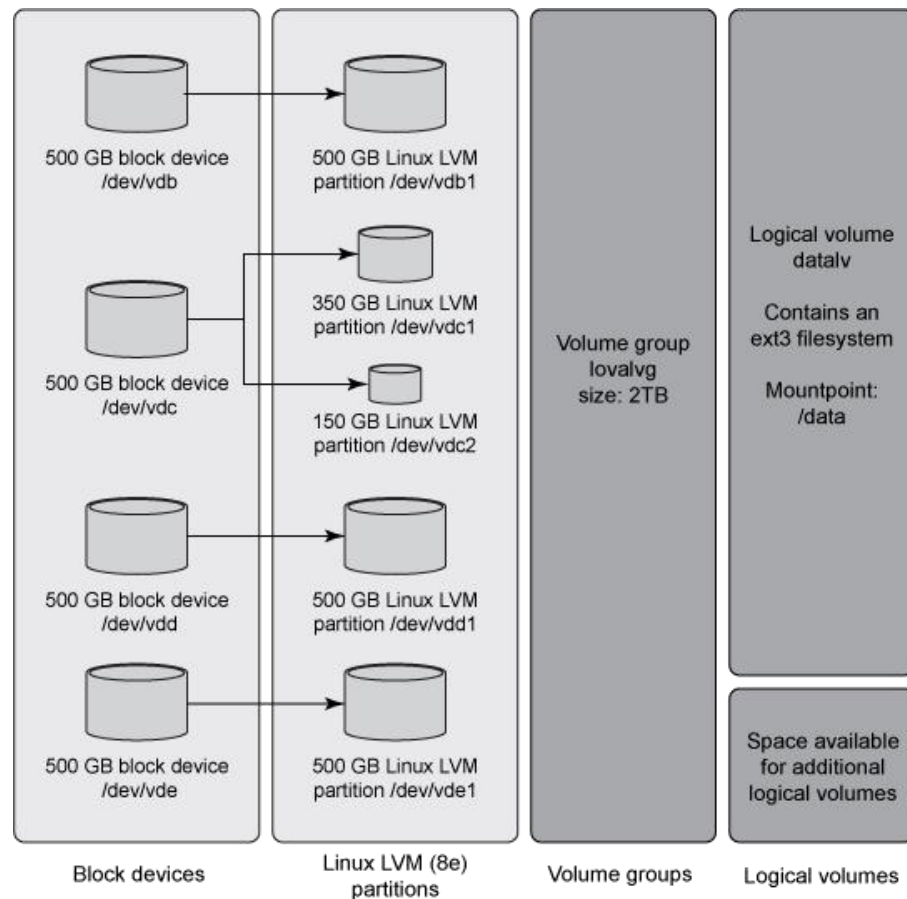
警告

使用 LVM 可能会增加一些风险，例如数据丢失。这些风险还包括应用程序崩溃、电源故障及有问题的命令。在实施 LVM 或重配置卷前，请保存数据。决不要在没有备份的情况下工作。

3.1. LVM 基本概念

Linux LVM 由物理卷（PV）、卷组（VG）和逻辑卷（LV）组成。

- **物理卷：**物理硬盘（HDD）、物理硬盘分区（例如 `/dev/vdb1`）。
- **扩展：**PV 被分为多个称为 PE（物理扩展）的块。逻辑扩展（LE）与 PE 一一对应，用于从物理到逻辑卷的映射。
- **卷组：**一个由聚合的物理卷组成的虚拟磁盘。VG 可以被逻辑分区为 LV。
- **逻辑卷：**充当一个虚拟磁盘分区。在创建完一个 VG 后，您可以在该 VG 中创建多个 LV。LV 可以用作原始块设备、交换设备，或者用于创建像磁盘分区一样（可挂载）的文件系统。
- **文件系统：**LV 可以用作原始设备或者交换设备，但更通常的是被“格式化”为某种受支持的文件系统，并挂载到确定的挂载点。
- **分区表：**您将使用一些工具（例如 `fdisk`、`sfdisk` 或者 `cgdisk`）来操作块设备分区表和创建 Linux LVM（8e）类型的分区。



3.2. 创建 LVM 分区

对于每个磁盘，将要用于 LVM 的可用空间分区为 0x8E Linux LVM。

```
[root@oe1db1 disk]# fdisk -l /dev/sda
```

```
Disk /dev/sda: 26.8 GB, 26843545600 bytes
255 heads, 63 sectors/track, 3263 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
```

Device	Boot	Start	End	Blocks	Id	System
/dev/sda1	*	1	13	104391	83	Linux
/dev/sda2		14	3263	26105625	8e	Linux LVM

Command (m for help): **t**

Selected partition 1

Hex code (type L to list codes): 8e

Changed system type of partition 1 to 8e (Linux LVM)

通过 **t** 参数，修改分区类型。修改后的类型，通过 **fdisk** 可以查看到

```
[root@oe1db1 rc.d]# fdisk -l /dev/sdf
```

Disk /dev/sdf: 2147 MB, 2147483648 bytes
255 heads, 63 sectors/track, 261 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

Device	Boot	Start	End	Blocks	Id	System
/dev/sdf1		1	261	2096451	8e	Linux LVM

否则分区后，默认的分类型始终为 83。

3.3. 配置物理卷 PV

将 Linux LVM 分区指派给卷组后，分区将称为物理卷。

```
[root@oe1db1 rc.d]# pvcreate /dev/sdd1
```

```
writing physical volume data to disk "/dev/sdd1"
Physical volume "/dev/sdd1" successfully created
```

```
[root@oe1db1 rc.d]# pvcreate /dev/sde1
```

```
writing physical volume data to disk "/dev/sde1"
Physical volume "/dev/sde1" successfully created
```

3.4. 创建卷组 VG

LVM 卷组将 Linux LVM 分区组织到逻辑空间池中。您可以从组的可用空间中划分出逻辑卷。

```
[root@oe1db1 rc.d]# vgcreate sky_vg /dev/sdd1 /dev/sde1
```

```
Volume group "sky_vg" successfully created
```

```
[root@oe1db1 rc.d]# vgs
```

VG	#PV	#LV	#SN	Attr	VSize	VFree
volGroup00	1	2	0	wz--n-	24.88G	0
sky_vg	2	0	0	wz--n-	1.98G	1.98G

3.5. 配置逻辑卷 LV

```
[root@oe1db1 rc.d]# lvcreate -L 1G -n lv_sky_01 sky_vg
```

```
Logical volume "lv_sky_01" created
```

```
[root@oe1db1 rc.d]# lvs
```

LV	VG	Attr	LSize	Origin	Snap%	Move	Log	Copy%	Convert
LogVol00	volGroup00	-wi-ao	20.97G						
LogVol01	volGroup00	-wi-ao	3.91G						
lv_sky_01	sky_vg	-wi-a-	1.00G						

3.6. 调整文件系统大小

```
[root@oe1db1 /]# umount /dev/sky_vg/lv_sky_01
```

```
[root@oe1db1 /]# e4fsck -f /dev/sky_vg/lv_sky_01
```

```
e4fsck 1.41.12 (17-May-2010)
```

```
Pass 1: Checking inodes, blocks, and sizes
```

```
Pass 2: Checking directory structure
```

```
Pass 3: Checking directory connectivity
```

```
Pass 4: Checking reference counts
```

```
Pass 5: Checking group summary information
```

```
/dev/sky_vg/lv_sky_01: 11/131072 files (0.0% non-contiguous), 35308/524288 blocks
```

```
[root@oe1db1 /]# resize4fs /dev/sky_vg/lv_sky_01
```

```
resize4fs 1.41.12 (17-May-2010)
```

```
Resizing the filesystem on /dev/sky_vg/lv_sky_01 to 1048576 (1k) blocks.
```

```
The filesystem on /dev/sky_vg/lv_sky_01 is now 1048576 blocks long.
```

```
[root@oe1db1 /]# mount /dev/sky_vg/lv_sky_01 /oradata/
```

```
[root@oeldb1 /]# df -h
Filesystem              Size  Used Avail Use% Mounted on
/dev/mapper/VolGroup00-LogVol00
                        21G   12G   8.3G  58% /
/dev/sda1                99M   24M   70M   26% /boot
tmpfs                   1004M  200M  805M   20% /dev/shm
/dev/sr0                 3.8G   3.8G    0 100% /media/disk
/dev/mapper/sky_vg-lv_sky_01
                        992M   19M  922M    2% /oradata
```

3.7. 常用命令

3.7.1. 扫描块设备 `lvmdiskscan`

```
[root@rhel61 rules.d]# lvmdiskscan | grep -v ram | grep /dev/sdc
/dev/sdc1                [      1.01 GiB]
/dev/sdc2                [    1011.91 MiB]
```

3.7.2. 物理卷

3.7.2.1 `pvdisplay` 显示一个物理卷的属性

```
[root@rhel61 ~]# pvdisplay
--- Physical volume ---
PV Name                /dev/sda2
VG Name                vg_rhel61
PV Size                29.51 GiB / not usable 3.00 MiB
Allocatable            yes (but full)
PE Size                4.00 MiB
Total PE               7554
Free PE                0
Allocated PE           7554
PV UUID                Yw00nE-ONCK-Ex9l-fkFu-7VA3-nLyy-dPn2Kd
```

3.7.2.2 `pvs` 报告有关物理卷的信息

```
[root@rhel61 ~]# pvs
PV          VG          Fmt Attr PSize PFree
/dev/sda2   vg_rhel61 lvm2 a-- 29.51g  0
```

3.7.3. 卷组

3.7.3.1 vgdisplay 显示卷组的属性

```
[root@rhel61 ~]# vgdisplay vg_rhel61
--- Volume group ---
VG Name                vg_rhel61
System ID
Format                 lvm2
Metadata Areas         1
Metadata Sequence No   3
VG Access               read/write
VG Status               resizable
MAX LV                 0
Cur LV                 2
Open LV                2
Max PV                 0
Cur PV                 1
Act PV                 1
VG Size                29.51 GiB
PE Size                4.00 MiB
Total PE                7554
Alloc PE / Size        7554 / 29.51 GiB
Free PE / Size          0 / 0
VG UUID                ib4arg-bmz0-i3gd-5ST3-gY4w-8kMN-Rfh03w
```

3.7.3.2 vgs 报告有关卷组的信息

```
[root@rhel61 ~]# vgs
VG          #PV #LV #SN Attr   VSize VFree
vg_rhel61   1   2   0 wz--n- 29.51g  0
```

3.7.4. 逻辑卷

3.7.4.1 lvdisplay 显示一个逻辑卷的属性

```
[root@rhel61 ~]# lvdisplay /dev/vg_rhel61/lv_root
--- Logical volume ---
LV Path                /dev/vg_rhel61/lv_root
LV Name                lv_root
VG Name                vg_rhel61
LV UUID                RYAAQH-44wV-ow4k-QNDG-g5Sy-U6tb-R0c4WH
```

```

LV Write Access      read/write
LV Creation host, time rhel61, 2013-11-29 11:30:28 +0800
LV Status            available
# open               1
LV Size              25.63 GiB
Current LE           6562
Segments             1
Allocation           inherit
Read ahead sectors   auto
- currently set to   256
Block device         253:0

```

3.7.4.2 lvs 报告有关逻辑卷的信息

```

[root@rhel61 ~]# lvs
  LV      VG      Attr      LSize  Pool Origin Data%  Move Log Cpy%Sync Convert
lv_root  vg_rhel61 -wi-ao--- 25.63g
lv_swap  vg_rhel61 -wi-ao---  3.88g

```

3.7.5. 检查元数据

3.7.5.1 pvck 检查物理卷元数据

```

[root@rhel61 ~]# pvck /dev/sda2
Found label on /dev/sda2, sector 1, type=LVM2 001
Found text metadata area: offset=4096, size=1044480

```

3.7.5.2 vgck 检查卷组元数据

```

[root@rhel61 ~]# vgck -v /dev/vg_rhel61
Using volume group(s) on command line
Finding volume group "vg_rhel61"

```

3.7.6. e2fsck

检查 Linux ext2/ext3/ext4 文件系统。

3.7.7. resize2fs

调整 ext2/ext3/ext4 文件系统的大小。

4. Veritas Volume Manager

Symantec 的 Veritas Volume Manager (VxVM) 是一个存储管理子系统，使用该系统可将物理磁盘作为一种称为“卷”的逻辑设备进行管理。VxVM 卷在应用程序与操作系统看来，就是可以在上面配置文件系统、数据库以及其他管理数据对象的物理磁盘分区设备。

4.1. Veritas Volume Manager 基本概念

4.1.1. VxVM 和操作系统

VxVM 充当操作系统与数据管理系统（如文件系统与数据库管理系统）之间的子系统。

VxVM 的下列功能与操作系统有关：

- 操作系统（磁盘）设备
- 设备句柄
- VxVM 动态多路径(DMP) 元设备

VxVM 依靠下列持续运行的后台驻留程序及其内核线程：

- vxconfigd —VxVM 配置后台驻留程序维护磁盘和磁盘组配置，将配置更改传递给内核，并修改磁盘上存储的配置信息。
- vxiod —VxVM I/O 内核线程提供扩展 I/O 操作，而不会阻止调用进程。默认情况下，引导时会启动 16 个 I/O 线程，且至少一个 I/O 线程将一直继续运行。
- vxrelocd — 热重定位后台驻留程序监视 VxVM 是否有影响冗余的事件，并执行热重定位以恢复冗余。

4.1.2. VxVM 如何进行存储管理

可以使用以下两种方法格式化硬盘和存储信息：物理存储布局和逻辑存储布局。VxVM 使用逻辑存储布局方法。

VxVM 使用以下两种对象进行存储管理：物理对象和虚拟对象。

- 物理对象 — 物理磁盘，或其他具有用于存储数据的块和原始操作系统设备接口的硬件。
- 虚拟对象 — 当有一个或多个物理磁盘被置于 VxVM 控制之下时，该软件将在这些物理磁盘上创建称为卷的虚拟对象。

4.1.2.1 物理对象 — 物理磁盘

物理磁盘是基本的存储设备（介质），即数据最终存储到的地方。可以使用设备名来定位物理磁盘，进而访问该磁盘上的数据。

典型设备名的形式为 **c#t#d#s#**，其中：

- **c#** 指定控制器
- **t#** 指定目标 ID
- **d#** 指定磁盘
- **s#** 指定分区或片

例如，设备名 **c0t0d0s2** 是与系统中 0 号控制器连接的整个硬盘，其目标 ID 为 0，物理磁盘号为 0。

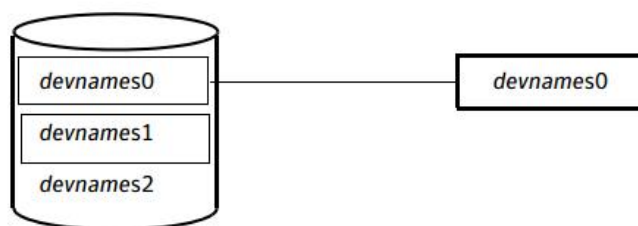
VxVM 在 VxVM 控制的物理磁盘（VM 磁盘）上写入标识信息。这样，即使在物理磁盘断开或系统断电后，也可以识别 VxVM 磁盘。VxVM 然后可以对磁盘组和逻辑对象进行重组，以便提供故障检测能力并加快系统修复速度。

➤ 分区

可以将物理磁盘划分成一个或多个分区（又称片）。然后在 **devname** 末尾加上分区号，用 **s#** 表示。

有多个分区的物理磁盘

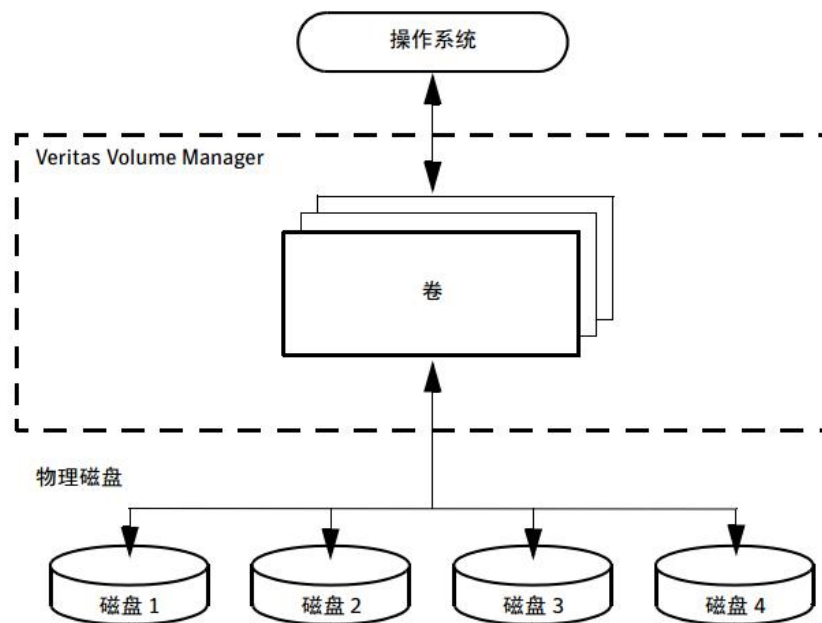
分区



➤ 磁盘阵列

磁盘阵列是物理磁盘的集合，VxVM 可将其作为一个或多个虚拟磁盘或卷提供给操作系统。

对于操作系统而言，VxVM 所创建的卷在外观和行为上均与物理磁盘类似。应用程序与卷交互的方式应和与物理磁盘交互的方式相同。



➤ 多路径处理磁盘阵列

有些磁盘阵列提供用于访问其磁盘设备的多个端口。这些端口与主机总线适配器(HBA) 控制器以及阵列的任何本地数据总线或 I/O 处理器相结合，组成了多个用于访问磁盘设备的硬件路径。这种磁盘阵列称为多路径处理磁盘阵列。

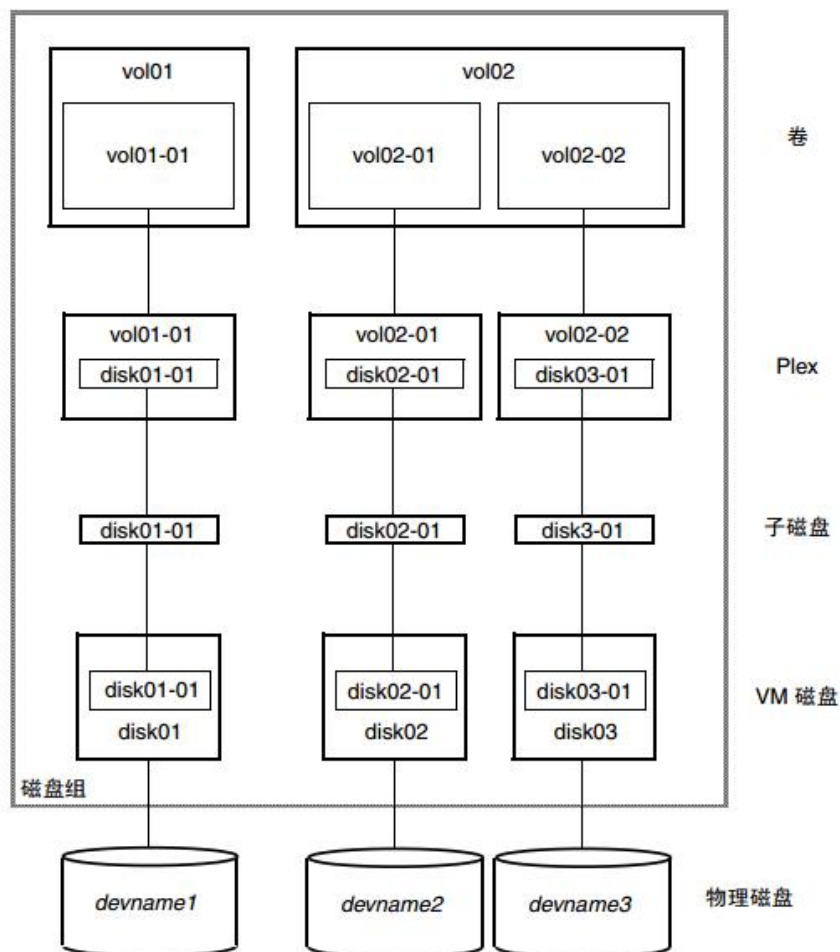
➤ 设备发现

设备发现服务与发现主机挂接的设备的能力相结合，使您能够动态地向新磁盘阵列添加支持。此操作（它使用一种称为设备发现层(DDL) 的功能）无须重新启动即可完成。

4.1.2.2 虚拟对象

可将 VxVM 的虚拟对象组合起来形成卷。卷中包含的虚拟对象有：VM 磁盘、磁盘组、子磁盘和 plex。Veritas Volume Manager 对象按如下方式组织：

- VM 磁盘可组成磁盘组
- 子磁盘（每个子磁盘代表磁盘的特定区域）可组合起来形成 plex
- 卷由一个或多个 plex 组成



➤ 磁盘组

磁盘组是使用公共配置并由 **VxVM** 管理的一些磁盘的集合。磁盘组配置是包含相关 **VxVM** 对象及其属性和联系的详细信息的记录集。磁盘组名称最长为 31 个字符。

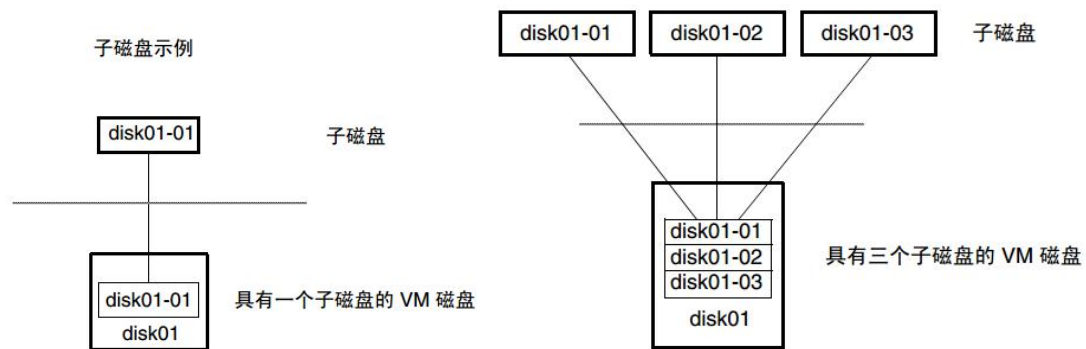
➤ VM 磁盘

将物理磁盘置于 **VxVM** 控制之下时，系统会为该物理磁盘分配一个 **VM 磁盘**。**VM 磁盘**受 **VxVM** 控制，通常位于某个磁盘组中。每个 **VM 磁盘**至少与一个物理磁盘或磁盘分区对应。

➤ 子磁盘

子磁盘是一组连续的磁盘块。块是磁盘上的空间单元。**VxVM** 使用子磁盘分配磁盘空间。

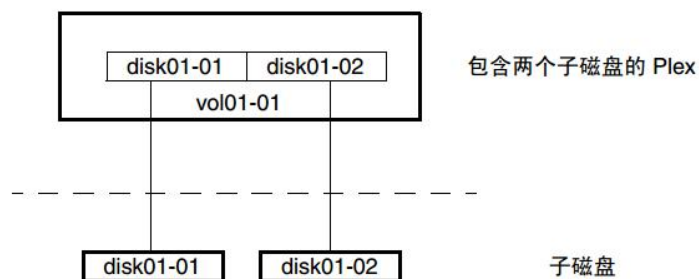
每个子磁盘代表一个 **VM 磁盘**的特定部分，而 **VM 磁盘**映射到物理磁盘的特定区域。



任何不属于子磁盘的 VM 磁盘空间都是空闲空间。空闲空间可用于创建新的子磁盘。

➤ Plex

VxVM 使用子磁盘创建称为 **plex** 的虚拟对象。**Plex** 由位于一个或多个物理磁盘上的一个或多个子磁盘组成。



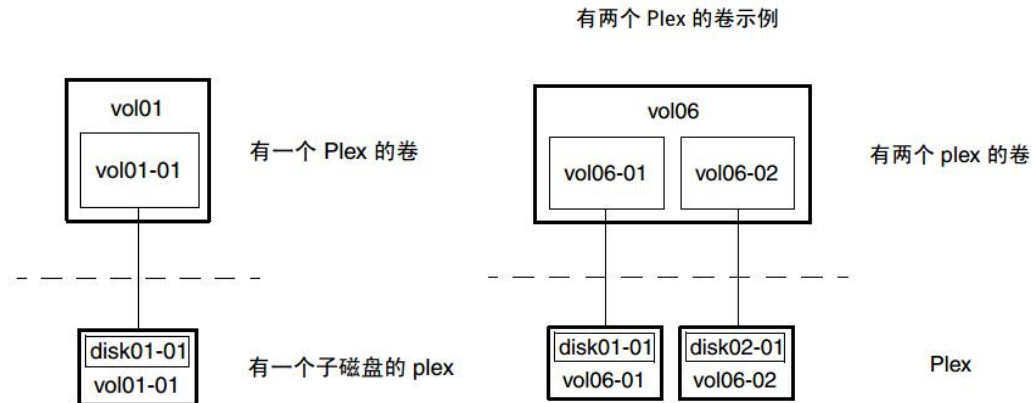
通过下列方法可以在子磁盘上组织数据以形成 **Plex**:

- 连续
- 条带化(RAID-0)
- 镜像(RAID-1)
- 带奇偶校验的条带化(RAID-5)

➤ 卷

卷是一种虚拟的磁盘设备，就应用程序、数据库和文件系统而言，它与物理磁盘设备很相似，但没有物理磁盘设备的物理局限性。

卷由一个或多个 **plex** 组成，每个 **plex** 都包含卷中所选数据的一个副本。



4.2. 管理磁盘

4.2.1. 磁盘设备

将磁盘置于 VxVM 控制之下时，系统会为其分配一个 VM 磁盘。

磁盘名最长可以包含 31 个字符。如果不分配磁盘名，则默认为 `diskgroup##`，其中 `diskgroup` 是将磁盘添加到的磁盘组的名称，`##` 是序列号。

设备名（有时称为 `devname` 或磁盘访问名）用于定义操作系统已知的磁盘设备的名称。这类设备通常（但并非总是）位于 `/dev/[r]dsk` 目录中。

VxVM 将在 `/dev/vx/[r]dmp` 目录中重新创建磁盘设备（包括 `/dev/[r]dsk` 目录下的那些磁盘设备）作为元设备。

可以使用 `vxdisk` 公用程序显示元设备包含的路径，并显示每条路径的状态（如启用还是禁用）。

4.2.1.1 VxVM 中的磁盘设备命名

➤ 基于 `c#t#d#s#` 命名

设备名的语法是 `c#t#d#s#`，其中 `c#` 表示主机总线适配器上的控制器，`t#` 表示目标控制器 ID，`d#` 代表目标控制器上的一个磁盘，`s#` 表示磁盘上的一个分区（或片）。

➤ 基于磁盘阵列的命名

设备名很长的设备（如包含全球名称(wwn)标识符的光纤通道设备）始终用基于磁盘阵列的名称表示。

要显示 VM 磁盘的本地 OS 设备名（如 `mydg01`），请使用以下命令：

```
# vxdisk path | egrep diskname
```

4.2.1.2 专用磁盘区域和公共磁盘区域

➤ 专用区域

存储配置信息的小块区域。磁盘头标签、**vxvm** 对象（如卷、**plex** 和子磁盘）的配置记录和配置数据库的意向日志都存储在这里。默认的专用区域的大小为 **32 MB**，足以记录磁盘组中数千个 **vxvm** 对象的详细信息。

➤ 公共区域

包括磁盘剩余空间的区域，用于为子磁盘分配存储空间。

4.2.2. 发现和配置新添加的磁盘设备

vxdiskconfig 公用程序扫描并配置挂接到主机的新磁盘设备、变为联机的磁盘设备或被分区到主机总线适配器（和此主机相连）的光纤通道设备。

每当磁盘以物理方式连接到主机或当光纤通道设备被分区到主机时，都应使用 **vxdiskconfig**。

vxdiskconfig 调用 **vxctl enable** 来重建卷设备节点的目录，并更新 **DMP** 内部数据库以反映系统的新情况。

还可以使用 **vxdisk scandisks** 命令扫描操作系统设备树中的设备，并启动多径处理磁盘的动态重新配置操作。

```
# vxctl -f enable
# vxdisk -f scandisks
```

如果进行如下修改因此而修改了系统配置，则启动完整扫描：

- 安装的阵列支持库。
- 列成被 **vxvm** 排除使用的设备。
- **DISKS**（**JBOD**）、**SCSI3** 或外来设备定义。

4.2.2.1 部分发现设备

下面的命令让 **vxvm** 发现以前不知道的新增设备：

```
# vxdisk scandisks new
```

下一个示例发现光纤设备：

```
# vxdisk scandisks fabric
```

下面的命令扫描设备 **c1t1d0** 和 **c2t2d0**：

```
# vxdisk scandisks device=c1t1d0,c2t2d0
```

或者，可以指定！前缀字符来表示要扫描除已列出的设备以外的所有设备：

```
# vxdisk scandisks !device=c1t1d0,c2t2d0
```

也可以扫描连接（或未连接）到一系列逻辑或物理控制器的设备：

```
# vxdisk scandisks !ctlr=c1,c2
```

下面的命令发现连接到指定物理控制器的设备：

```
# vxdisk scandisks pctlr=/pci@1f,4000/scsi@3/
```

4.2.3. 发现磁盘和动态添加磁盘阵列

4.2.3.1 添加对新磁盘阵列的支持

```
# pkgadd -d /cdrom/pkgdir/vrtsda
```

➤ 启用新设备发现功能

要让 **VxVM** 发现新的磁盘阵列，请使用以下命令：

```
# vxdctl enable
```

➤ 删除对磁盘阵列的支持

要删除对 **vrtsda** 磁盘阵列的支持，请使用以下命令：

```
# pkgrm vrtsda
```

➤ 列出支持的磁盘阵列的细节

要列出当前支持的所有磁盘阵列，请使用以下命令：

```
# vxddladm listsupport all
```

要显示特定阵列库的详细信息，请使用以下命令形式：

```
# vxddladm listsupport libname=library_name.so
```

➤ 列出被拒绝的磁盘阵列

要列出 **VxVM** 当前拒绝使用的所有磁盘阵列，请使用以下命令：

```
# vxddladm listexclude
```

4.2.4. 将磁盘置于 VxVM 控制之下

4.3. 常用命令

4.3.1. 获取关于 VxVM 对象的信息

- 列出 VxVM 的许可功能

```
vxctl license
```

- 列出有关磁盘组的信息

```
vxvg list [diskgroup]
```

- 列出有关共享磁盘组的信息

```
vxvg -s list
```

- 显示有关卷的可访问性和可用性的信息

```
vxinfo [-g diskgroup] [volume...]
```

- 打印有关 VxVM 中对象的单行信息

```
vxprint -hrt [-g diskgroup] [object]
```

- 显示有关子磁盘的信息

```
vxprint -st [-g diskgroup] [subdisk]
```

- 显示有关 plex 的信息

```
vxprint -pt [-g diskgroup] [plex]
```

4.3.2. 管理磁盘

4.3.3. 创建和管理磁盘组

4.3.4. 创建和管理子磁盘

4.3.5. 创建和管理 Plex

4.3.6. 创建卷

4.3.7. 监视和控制任务

5. 5