资料序号		产品名称	S5100/S1200/S3200
使用对象	华为工程师、合作方工程师	产品版本	V3R1/V2
编写部门	系统集成及技术服务部	资料版本	V1.0

Linux LVM 卷管理操作指导书

拟　制：	曹钢	日　期：	2007-11-30
审　核：		日　期：	
审　核：		日　期：	
批　准：		日　期：	

华 为 技 术 有 限 公 司

修订记录

日期	修订版本	描述	作者
2007-11-30 1.0		完成初稿	曹钢

关键词：

LVM 逻辑卷 pv vg lv

描述：

LVM 是逻辑盘卷管理（ Logical Volume Manager ）的简称，它是 Linux 环境下对磁盘分区进行管理的一种机制， LVM 是建立在硬盘和分区之上的一个逻辑层，来提高磁盘分区管理的灵活性。通过 LVM 系统管理员可以轻松管理磁盘分区，如：将若干个磁盘分区连接为一个整块的卷组（ volume group ），形成一个存储池。管理员可以在卷组上随意创建逻辑卷组（ logical volumes ），并进一步在逻辑卷组上创建文件系统。管理员通过 LVM 可以方便的调整存储卷组的大小，并且可以对磁盘存储按照组的方式进行命名、管理和分配。

相关命令：

pvcreate vgcreate vgchange lvcreate mkfs.X pvresize pvscan vgscan lvscan lvextend e2fsck resize2fs

问题简述：

由于目前我公司大多数存储提供的 lun 的最大容量为 2T，此容量有时不满足用户大容量需求，故需要用 LVM 管理重新规划较大容量的 lun，且此规划的 lun 还可以自由调配。

缩略语清单：

LVM	Logical Volume Manager(逻辑卷管理)
pv	物理卷（ physical volume ）
Vg	卷组（ Volume Group ）
lv	逻辑卷（ logical volume ）

参考手册：

LVM HowTo 提供一份比较基础的 LVM 指南。

在 LVM2 工具的 Linux 手册页上可以找到更多细节。

目 录

第1章 LVM 简介

1.1 前言

每个 Linux 使用者在安装 Linux 时 都会遇到这样的困境： 在为系统分区时，如何精确评估和分配各个硬盘分区的容量，因为系统管理员不但要考虑到当前某个分区需要的容量，还要预见该分区以后可 能需要的容量的最大值。因为如果估计不准确，当遇到某个分区不够用时管理员可能甚至要备份整个系统、清除硬盘、重新对硬盘分区，然后恢复数据到新分区。

虽然现在有很多动态调整磁盘的工具可以使用，例如 Partation Magic 等等，但是它并不能完全解决问题，因为某个分区可能会再次被耗尽；另外一个方面这需要重新引导系统才能实现，对于很多关键的服务器，停机是不可接受的，而且对于添加新硬盘，希望一个能跨越多个硬盘驱动器的文件系统时，分区调整程序就不能解决问题。

因此完美的解决方法应该是在零停机前提下可以自如对文件系统的大小进行调整，可以方便实现文件系统跨越不同磁盘和分区。幸运的是 Linux 提供的逻辑盘卷管理（ LVM， Logical Volume Manager ）机制就是一个完美的解决方案。

1.2 什么是 LVM

LVM 是 Logical Volume Manager(逻辑卷管理)的简写，它由 Heinz Mauelshagen 在 Linux 2.4 内核上实现， 目前最新版本为： 稳定版 1.0.5 , 开发版 1.1.0-rc2 , 以及 LVM2 开发版。与 传统的磁盘与分区相比， LVM 为计算机提供了更高层次的磁盘存储。它是 Linux 环境下对磁盘分区进行管理的一种机制， LVM 是建立在硬盘和分区之上的一个逻辑层， 来提高磁盘分区管理的灵活性。通过 LVM 系统管理员可以轻松管理磁盘分区，如：将若干个磁盘分区连接为一个整块的卷组（ volume group ），形成一个存储池。管理员可以在卷组上随意创建逻辑卷组（ logical volumes ），并进一步在逻辑卷组上创建文件系统。 管理员通过 LVM 可以方便的调整存储卷组的大小，并且可以对磁盘存储按照组的方式进行命名、管理和分配，例如按照使用用途进行定义： “ development ”和“ sales, ”而不是使用物理磁盘名 “ sda”和“ sdb。”而且当系统添加了新的磁盘，通过 LVM 管理员就不必将磁盘的文件移动到新的磁盘上以充分利用新的存储空间，而是直接扩展文件系统跨越磁盘即可。

1.3 为什么使用 LVM

LVM 通常用于装备大量磁盘的系统，但它同样适于仅有一、两块硬盘的小系统。

1.3.1 小系统使用 LVM 的益处

传统的文件系统是基于分区的，一个文件系统对应一个分区。这种方式比较直观，但不易改变：

(1) 不同的分区相对独立，无相互联系，各分区空间很易利用不平衡，空间不能充分利用；

(2) 当一个文件系统／分区已满时，无法对其扩充，只能采用重新分区／建立文件系统，非常麻烦；或把分区中的数据移到另一个更大的分区中；或采用符号连接的方式使用其它分区的空间。

(3) 如果要把硬盘上的多个分区合并在一起使用，只能采用再分区的方式，这个过程需要数据的备份与恢复。

当采用 LVM 时，情况有所不同：

(1) 硬盘的多个分区由 LVM 统一为卷组管理，可以方便的加入或移走分区以扩大或减小卷组的可用容量，充分利用硬盘空间；

(2) 文件系统建立在逻辑卷上，而逻辑卷可根据需要改变大小(在卷组容量范围内)以满足要求；

(3) 文件系统建立在 LVM 上，可以跨分区，方便使用；

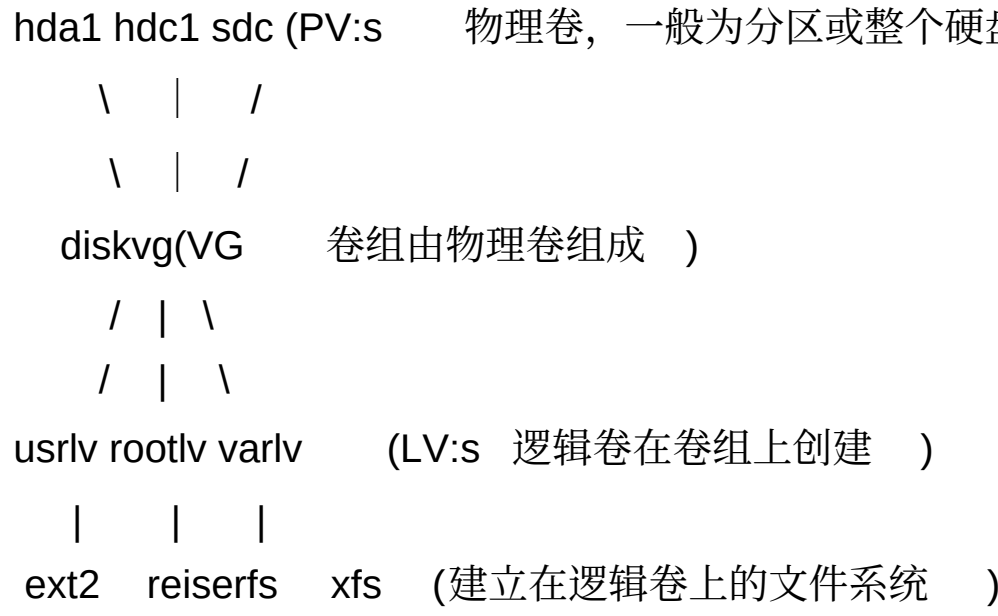
1.3.2 大系统使用 LVM 的益处

在使用很多硬盘的大系统中，使用 LVM 主要是方便管理、增加了系统的扩展性。

在一个有很多不同容量硬盘的大型系统中，对不同的用户的空间分配是一个技巧性的工作，要在用户需求与实际可用空间中寻求平衡。用户／用户组的空间建立在 LVM 上，可以随时按要求增大，或根据使用情况对各逻辑卷进行调整。当系统空间不足而加入新的硬盘时，不必把用户的数据从原硬盘迁移到新硬盘，而只须把新的分区加入卷组并扩充逻辑卷即可。同样，使用 LVM 可以在不停服务的情况下。把用户数据从旧硬盘转移到新硬盘空间中去。

第2章 LVM 构成

2.1 LVM 的结构简图



2.2 LVM 的组成

1. 卷组 volume group (VG)

卷组是 LVM 中最高抽象层, 是由一个或多个物理卷所组成的存储器池。

2. 物理卷 physical volume (PV)

典型的物理卷是硬盘分区，但也可以是整个硬盘或已创建的 Software RAID 卷。

3. 逻辑卷 logical volume (LV)

逻辑卷相当于非 LVM 系统中的分区，它在卷组上建立，是一个标准的块设备，可以在其上建立文件系统。

4. 物理块 physical extent (PE)

物理卷按大小相等的 “块 ”为单位存储，块的大小与卷组中逻辑卷块的大小相同。

5. 逻辑块 logical extent (LE)

逻辑卷按 “块 ”为单位存储，在一卷组中的所有逻辑卷的块大小是相同的。

6. 总述

例 子：有一个卷组 VG1，它的物理块大小为 4MB。在这个卷组中为 2 个硬盘分区： /dev/hda1 与/dev/hdb1 ，它们分别成为物理卷 PV1 与 PV2 。物理卷将按 4MB 为单位分块，如 PV1 与 PV2 分别可分为 99 与 248 块。在 VG1 上建立逻辑卷，它的大小可在 1 至

347(99+248) 块之 间。当建立逻辑卷时，会建立逻辑块与物理块的一一映射关系。

7. 映射模式 (linear/striped)

在建立逻辑卷时，可以选择逻辑块与物理块映射的策略：

(1) 线性映射－将把一定范围的物理块按顺序分配给逻辑卷， 如 LV 的 LE 1 － 99映射到 PV1，LE 100 － 347映射到 PV2。

(2) 交错模式－将把逻辑块交错映射到不同的物理卷中， 如 LV 的LE 1 映射为 PV1 的PE1，LE 2 映射为 PV2 的PE1，LE 3 映射为 PV1 的 PE2...。这种方式可以提高逻辑卷的性能，但是采用这种方式建立的逻辑卷将不能在它们所在的物理卷中扩展。

8. Snapshots (快照)

LVM 提供了一个非常好的特性： snapshots 。它允许管理员建立一个块设备：该设备是一逻辑卷在某一时刻冻结的精确拷贝。这个特性通常用于批处理过程 (如 备份) 需要处理逻辑卷，但又不能停止系统。当操作完成时， snapshot 设备可以被移除。这个特性要求在建立 snapshot 设备时逻辑卷处于相容状态。

2.3 LVM 的安装及启动

(1) 首先确定系统中是否安装了 lvm 工具：

```
[root@www root]# rpm －qa|grep lvm
lvm-1.0.3-4
```

如果命令结果输入类似于上例，那么说明系统已经安装了 LVM 管理工具；如果命令没有输出则说明没有安装 LVM 管理工具，则需要从网络下载或者从光盘装 LVM rpm 工具包。

(2) 安装了 LVM 的 RPM 软件包以后， 要使用 LVM 还需要配置内核支持 LVM。RedHat 默认内核是支持 LVM 的，如果需要重新编译内核，则需要在配置内核时，进入 Multi-device Support (RAID and LVM) 子菜单，选中以下两个选项：

```
[*] Multiple devices driver support (RAID and LVM)
<*> Logical volume manager (LVM) Support
```

然后重新编译内核，即可将 LVM 的支持添加到新内核中。

(3) 为了使用 LVM，要确保在系统启动时激活 LVM，幸运的是在 RedHat7.0 以后的版本，系统启动脚本已经具有对激活 LVM 的支持，在 /etc/rc.d/rc.sysinit 中有以下内容：

```
# LVM initialization
if [ -e /proc/lvm -a -x /sbin/vgchange -a -f /etc/lvmtab ]; then
action $"Setting up Logical Volume Management:" /sbin/vgscan &&
/sbin/vgchange -a y
fi
```

其中关键是两个命令， vgscan 命令实现扫描所有磁盘得到卷组信息，并创建文件卷组数据文件 /etc/lvmtab 和/etc/lvmtab.d/* ；vgchange -a y 命令激活系统所有卷组。

(4) suse9 sp3 中 lvm 的启动方式

"为使系统启动时可自动激活并使用 LVM，可将以下几行添加到启动 rc 脚本中：

vi /etc/init.d/rc 在末尾加上下列命令：

```
/sbin/vgscan
/sbin/vgchange -a y
```

这些行将浏览所有可用的卷组并激活它们。要注意的是，它们应在安装卷组上的文件系统操作之前被执行，否则将无法正常安装文件系统。

第3章 LVM 的一般操作

3.1 修改磁盘的分区格式为 lvm 格式 8e

查看硬盘空间，并创建分区 (按以下命令逐行执行)

fdisk -l

fdisk /dev/sdb （如果每个 lun 一个分区，就可以不新建分区，只是修改分区格式 ）

p 查看当前分区情况

n 新建分区

p 主分区

2 建第 2 个主分区 (根据实际情况)

+20G 设置分区大小

t 设置分区格式

8e 设置为 linux lvm 格式

w 保存设置

q 退出分区

3.2 建立 PV

为把一个磁盘或分区作为 PV，首先应使用 pvcreate 对其初始化，如对 IDE 硬盘 /dev/hdb ，

"使用整个磁盘,

pvcreate /dev/hdb

这将在磁盘上建立 VG 的描述符。

"使用磁盘分区，如 /dev/hdb1 。

使用 fdisk 的 t 命令把 /dev/hda1 的分区类型设为 0x8e ，然后运行：

pvcreate /dev/hdb1

这将在分区 /dev/hda1 上建立 VG 的描述符。

PV 初始化命令 pvcreate 的一般用法为：

pvcreate PV1 [PV2 ...]

它的参数可以是整个磁盘、分区，也可以是一 loop 设备。

3.3 建立 VG

在使用 pvcreate 建立了 PV 后，可以用 vgcreate 建立卷组，如有 PV1、PV2 分别是 /dev/hda1 与/dev/hdb1 ，使用

vgcreate testvg /dev/hda1 /dev/hdb1

将建立一个名为 testvg 的卷组，它由两个 PV：/dev/hda1 与/dev/hdb1 组成。vgcreate 的一般用法为：

vgcreate [options] VG_name PV1 [PV2 ...]

其中的可选项包括设置 VG 最大支持的 LV 数、PE 大小 (缺省为 4MB) 等。

注意：当使用 devfs 系统时，应使用设备的全名而不能是 Symbol Link ，如对上例应为：

vgcreate testvg /dev/ide/host0/bus0/target0/lun0/part1\
/dev/ide/host0/bus0/target1/lun0/part1

3.4 激活 VG

在被激活之前， VG 与 LV 是无法访问的，这时可用命令：

vgchange -a y testvg

激活所要使用的卷组。当不再使用 VG 时，可用

vgchange -a n testvg

使之不再可用。

vgchange 可用来设置 VG 的一些参数，如是否可用 (-a [y|n] 选项)、支持最大逻辑卷数等。

3.5 移除 VG

在移除一卷组前应确认卷组中不再有逻辑卷，首先休眠卷组：

vgchange -a n testvg

然后可用 vgremove 移除该卷组：

vgremove testvg

3.6 为 VG 增加新 PV

当卷组空间不足时，可以加入新的物理卷来扩大容量，这时可用命令 vgextend ，如

vgextend testvg /dev/hdc1

其中 /dev/hdc1 是新的 PV，当然在这之前，它应使用 pvcreate 初始化。

3.7 从 VG 移除 PV

在移除 PV 之前，应确认该 PV 没用被 LV 使用，这可用命令 pvdisplay 查看，如：

```
# pvdisplay /dev/hda1
--- Physical volume ---
PV Name                /dev/hda1
VG Name                testvg
PV Size                1.95 GB / NOT usable 4 MB [LVM: 122 KB]
PV#                    1
PV Status              available
Allocatable            yes (but full)
Cur LV                 1
PE Size (KByte)        4096
Total PE               499
Free PE                0
Allocated PE           499
PV UUID                Sd44tK-9IRw-SrMC-MOkn-76iP-iftz-OVSen7
```

如这个 PV 仍在被使用，则应把数据传移到其它 PV 上。在确认它未被使用后，可用命令 vgreduce 把它从 VG 中删除，如：

```
# vgreduce testvg /dev/hda1
```

3.8 创建 LV

在创建逻辑卷前，应决定 LV 使用哪些 PV，这可用命令 vgdisplay 与 pvdisplay 查看当前卷组与 PV 的使用情况。在已有的卷组上创建逻辑卷使用命令 lvcreate，如：

```
# lvcreate -L1500 -ntestlv testvg
```

将在卷组 testvg 上建立一个 1500MB 的线性 LV，其命名为 testlv，对应的块设备为 /dev/testvg/testlv 。

```
# lvcreate -i2 -I4 -l100 -nanothertestlv testvg
```

将在卷组 testvg 上建立名为 anothertestlv 的 LV，其大小为 100LE，采用交错方式存放，交错值为 2，块大小为 4KB 。

如果需要 LV 使用整个 VG，可首先用 vgdisplay 查找 Total PE 值，然后在运行 lvcreate 时指定，如：

```
# vgdisplay testvg | grep "Total PE"
Total PE               10230
# lvcreate -l 10230 testvg -n mylv
```

将使用卷组 testvg 的全部空间创建逻辑卷 mylv 。

在创建逻辑卷后，就可在其上创建文件系统并使用它。

命令 lvcreate 的常用方法：

lvcreate [options] -n 逻辑卷名 卷组名 [PV1 ...]

其中的常用可选项有：

"-i Stripes ：采用交错 (striped) 方式创建 LV，其中 Stripes 指卷组中 PV 的数量。

"-I Stripe_size ：采用交错方式时采用的块大小 (单位为 KB)，Stripe_size 必须为 2 的指数： 2N ， N=2,3...9 。

"-l LEs ：指定 LV 的逻辑块数。

"-L size ：指定 LV 的大小，其后可以用 K、 M、 G 表示 KB、 MB、 GB。

"-s ：创建一已存在 LV 的 snapshot 卷。

"-n name ：为 LV 指定名称。

3.9 删除 LV

为删除一个逻辑卷，必须首先从系统卸载其上的文件系统，然后可用 lvremove 删除，如：

```
# umount /dev/testvg/testlv
# lvremove /dev/testvg/testlv
lvremove -- do you really want to remove "/dev/testvg/testlv"? [y/n]: y
lvremove -- doing automatic backup of volume group "testvg"
lvremove -- logical volume "/dev/testvg/testlv" successfully removed
```

3.10 扩展 LV

1. 例子：

为逻辑卷增加容量可用使用 lvextend ，即可以指定要增加的尺寸也可以指定扩容后的尺寸，如

```
# lvextend -L12G /dev/testvg/testlv
lvextend -- extending logical volume "/dev/testvg/testlv" to 12 GB
lvextend -- doing automatic backup of volume group "testvg"
lvextend -- logical volume "/dev/testvg/testlv" successfully
extended
```

将扩大逻辑卷 testlv 的容量为 12GB 。

```
# lvextend -L+1G /dev/testvg/testlv
lvextend -- extending logical volume "/dev/testvg/testlv" to 13 GB
lvextend -- doing automatic backup of volume group "testvg"
lvextend -- logical volume "/dev/testvg/testlv" successfully
extended
```

将为 LV testlv 再增大容量 1GB 至 13GB 。

为 LV 扩容的一个前提是： LV 所在的 VG 有足够的空闲存储空间可用。

在为 LV 扩容之后，应同时为 LV 之上的文件系统扩容，使二者相匹配。对不同的文件系统有相对应的扩容方法。

2. ext2/ext3

除非内核已有 ext2online 补丁，否则在改变 ext2/ext3 文件系统的大小时应卸载它：

```
# umount /dev/testvg/testlv
# resize2fs /dev/testvg/testlv
# mount /dev/testvg/testlv /home
```

这里假设 testlv 安装点为 /home 。在 es2fsprogs-1.19 或以上版本中包含 resize2fs 命令。

在 LVM 发行包中有一个称为 e2fsadm 的工具，它同时包含了 lvextend 与 resize2fs 的功能，如：

```
# e2fsadm -L+1G /dev/testvg/testlv
```

等价于下面两条命令：

```
# lvextend -L+1G /dev/testvg/testlv
# resize2fs /dev/testvg/testlv
```

但用户仍需首先卸载文件系统。

3. reiserfs

与 ext2 不同， Reiserfs 不必卸载文件系统，如：

```
# resize_reiserfs -f /dev/testvg/testvl
```

4. xfs

SGI XFS 文件系统必须在安装的情况下才可改变大小， 并且要使用安装点而不是块设备，如：

```
# xfs_growfs /home
```

3.11 缩小 LV

逻辑卷可扩展同样也可缩小，但应在缩小 LV 之前首先减小文件系统，否则将可能导致数据丢失。

1. ext2/ext3

可以使用 LVM 的工具 e2fsadm 操作，如：

```
# umount /home
# e2fsadm -L-1G /dev/testvg/testvl
# mount /home
```

如果采用 resize2fs ，就必须知道缩少后卷的块数：

```
# umount /home
# resize2fs /dev/testvg/testvl 524288
# lvreduce -L-1G /dev/testvg/testvl
# mount /home
```

2. reiserfs

在缩小 reiserfs 时，应首先卸载它，如：

```
# umount /home
# resize_reiserfs -s-1G /dev/testvg/testvl
# lvreduce -L-1G /dev/testvg/testvl
# mount -treiserfs /dev/testvg/testvl /home
```

3. xfs

无法实现。

3.12 在 PV 间转移数据

若要把一个 PV 从 VG 中移除，应首先把其上所有活动 PE 中的数据转移到其它 PV 上，而新的 PV 必须是本 VG 的一部分，有足够的空间。

如要把 PV1:/dev/hda1 上的数据移到 PV2:/dev/sda1 上可用命令：

```
# pvmove /dev/hdb1 /dev/sdg1
```

如果在该 PV 之上的 LV 采用交错方式存放，则这个转移过程不能被打断。

建议在转移数据之前备份 LV 中的数据。

3.13 系统启动／关闭 LVM

"为使系统启动时可自动激活并使用 LVM ，可将以下几行添加到启动 rc 脚本中：

```
/sbin/vgscan
/sbin/vgchange -a y
```

这些行将浏览所有可用的卷组并激活它们。要注意的是，它们应在安装卷组上的文件系统操作之前被执行，否则将无法正常安装文件系统。

"在系统关机时，要关闭 LVM，这可将以下这行添加到关机 rc 脚本中，并确保它在卸装了所有文件系统后执行：

/sbin/vgchange -a n

第4章 磁盘分区问题

4.1 一个磁盘上的多个分区

1. LVM 允许 PV 建立在几乎所有块设备上，如整个硬盘、硬盘分区、Soft RAID ：

```
# pvcreate /dev/sda1
# pvcreate /dev/sdf
# pvcreate /dev/hda8
# pvcreate /dev/hda6
# pvcreate /dev/md1
```

所以在一块硬盘上可以有多个 PV ／分区，但一般建议一块硬盘上只有一个 PV ：

"便于管理，易于处理错误 "避免交错方式中性能下降。 LVM 不能辨别两个 PV 是否在同一硬盘上，故当采用交错方式时，会导致性能更差。

但在某些情况下可采用： "把已存在的系统合并到 LVM 中。在一个只有少数硬盘的系统中，转换为 LVM 时需在在各分区之间转移数据。 "把一个大硬盘分给不同的 VG 使用。当一个 VG 的有不同的 PV 在同一硬盘时，创建交错方式的 LV 时应注意使用哪一个 PV。

2. Sun disk labels

仅在 SUN 的 SPARC 系统中有此问题。

第5章 建立 LVM 用例

在本节中，将在 3 块 SCSI 硬盘： /dev/sda ， /dev/sdb ， /dev/sdc 上按步建立 LVM 。

5.1 修改分区格式为 lvm 分区

fdisk /dev/sda （如果每个 lun 一个分区，就可以不新建分区，只是修改分区格式 ）

t 设置分区格式

8e 设置为 linux lvm 格式

w 保存设置

q 退出分区

5.2 准备分区

首先要做的是初始化硬盘，建立 PV ，这将会删除硬盘上的原有数据。

在此，用整个硬盘为 PV ：

```
# pvcreate /dev/sda
# pvcreate /dev/sdb
# pvcreate /dev/sdc
```

pvcreate 在每个硬盘的起始端建立卷组描述区(volume group descriptor area, VGDA) 。

5.3 创建卷组

利用上面三个 PV 建立卷组：

```
# vgcreate test_vg /dev/sda /dev/sdb /dev/sdc/
```

然后可用 vgdisplay 查看／验证卷组的信息：

```
# vgdisplay
--- Volume Group ---
VG Name                test_vg
VG Access              read/write
VG Status              available/resizable
VG #                   1
MAX LV                 256
Cur LV                 0
```

Open LV 0
MAX LV Size 255.99 GB
Max PV 256
Cur PV 3
Act PV 3
VG Size 1.45 GB
PE Size 4 MB
Total PE 372
Alloc PE / Size 0 / 0
Free PE / Size 372/ 1.45 GB
VG UUID nP2PY5-5TOS-hLx0-FDu0-2a6N-f37x-0BME0Y
其中最重要的前三条要正确，且 VS size 是以上三个硬盘容量之和。

5.4 建立 LV

在确定卷组 test_vg 正确后，就可在其上创建 LV。LV 的大小可在 VG 大小范围内任意选择，如同在硬盘上分区。

1. 建立线性方式 LV

在 test_vg 上建立一个大小为 1GB 的线性方式 LV：

```
# lvcreate －L 1G －n test_lv test_vg
lvcreate -- doing automatic backup of "test_vg"
lvcreate -- logical volume "/dev/test_vg/test_lv" successfully created
```

2. 建立交错方式 LV

在 test_vg 上建立一个大小为 1GB 的交错方式 LV，交错参数为 4KB ：

```
# lvcreate -i3 -I4 -L1G -ntest_lv test_vg
lvcreate -- rounding 1048576 KB to stripe boundary size 1056768 KB /
258 PE
lvcreate -- doing automatic backup of "test_vg"
lvcreate -- logical volume "/dev/test_vg/test_lv" successfully created
```

注意：如果使用 -i2 参数，则 LV 将仅使用 test_vg 中的两块硬盘。

5.5 建立文件系统

在 LV test_lv 创建后，就可在其上建立文件系统，

如， ext2/ext3 系统：

```
# mke2fs /dev/test_vg/test_lv
```

如， reiserfs ：

```
# mkreiserfs /dev/test_vg/test_lv
```

5.6 测试文件系统

安装 LV：

mount /dev/test_vg/test_lv /mnt

df

Filesystem 1k-blocks Used Available Use% Mounted on

/dev/hda1 1311552 628824 616104 51% /

/dev/test_vg/test_lv 1040132 20 987276 0% /mnt

则可以通过 /mnt 访问 LV。

5.7 Suse9 下的 lvm 详细配置过程

1. 版本：

服务器 suse linux 9 sp3 2.6.5 －7.244 －smp

服务器系统自带 LVM 2.01.14

华为存储 OceanStor S5100

2. 组网概叙：

S5100 规划了 19 个 lun 给服务器，每个 lun 都是 2T，总共 38T 容量。服务器已经可以应用 19 个 lun，但是用户要求在 linux 系统中用 LVM 重新规划 lun，每个 lun 要求 8T（linux 系统及 XFS 的文件系统都支持 16T），最后规划 4 个 8T 的 lv 和 1 个 6T 的 lv。1、用 fdisk －

3. 配置步骤：

(1) 查看所有 S5100 分配的 lun

/dev/sdb

/dev/sdc

/dev/sdd

/dev/sde

/dev/sdf

/dev/sdg

/dev/sdh

/dev/sdi

/dev/sdg

/dev/sdk

/dev/sdl

/dev/sdm

/dev/sdn

/dev/sdo

/dev/sdp

/dev/sdq

/dev/sdr

/dev/sds

/dev/sdt

(2) 在硬盘上创建一个 LVM 分区。使用 fdisk 或者其它的分区工具来创建一个 LVM 分区。 Linux LVM 的分区类型为 8e。

fdisk /dev/sdb

t

8e

w

根据以上步骤，修改所以的要用来做 pv 的 lun

(3) 创建 pv。下述命令将在分区的起始处创建一个卷组描述符：

pvcreate /dev/sdb

pvcreate -- -physical volume "/dev/sdb" successfully created

可以用以下命令创建所有的 pv

pvcreate /dev/sd{c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t}

可以使用 pvs 检查

(4) 创建卷组 VG。通过下面的方法创建一个新的卷组，并且添加两个物理卷：

vgcreate vg0 /dev/sdb /dev/sdc

vgcreate- -- volume group "vg0" successfully created and activated

上述命令将创建一个名为 vg0 的卷组，包含有 /dev/sdb 到/dev/sdt19 个物理卷的卷组。

(5) 使用下面命令来激活卷组：

vgchange -ay vg0

可以使用 vgs 检查，使用“ vgdisplay ”命令来查看所建立卷组的细节信息。

(6) 创建逻辑卷。使用 lvcreate 命令在卷组中创建一个逻辑卷：

lvcreate -L 8T -n lv1 vg0

lvcreate -L 8T -n lv2 vg0

lvcreate -L 8T -n lv3 vg0

lvcreate -L 8T -n lv4 vg0

lvcreate -L 6T -n lv5 vg0

可以使用 lvs 检查

(7) 创建文件系统。在该逻辑卷上选择使用 XFS 文件系统：

对于视频存储应用，研发建议使用 XFS 文件系统

sunit ：分条深度，以扇区为单位， 64k 分条深度， sunit 为 128

swidth ：sunit 乘以数据盘数

mkfs.xfs －f －d sunit=128,swidth=2304 /dev/vg0/lv1

mkfs.xfs －f －d sunit=128,swidth=2304 /dev/vg0/lv2

mkfs.xfs －f －d sunit=128,swidth=2304 /dev/vg0/lv3

mkfs.xfs －f －d sunit=128,swidth=2304 /dev/vg0/lv4

mkfs.xfs －f －d sunit=128,swidth=2304 /dev/vg0/lv5

(8) 使用 mount 命令来加载新创建的文件系统。（ recordfile1 ， recordfile2 ， recordfile3 ， recordfile4 ， recordfile5 为新建的 mount 点）

```
# mount -t xfs /dev/vg0/lv1 /recordfile1
# mount -t xfs /dev/vg0/lv2 /recordfile2
# mount -t xfs /dev/vg0/lv3 /recordfile3
# mount -t xfs /dev/vg0/lv4 /recordfile4
# mount -t xfs /dev/vg0/lv5 /recordfile5
```

(9) 在 /etc/fstab 中加入以下入口，在启动时加载文件系统：（如果不成功，建议用编写自动 mount 脚本执行，且加到启动的 rc 末尾执行）

```
/dev/vg0/lv1 /recordfile1 xfs defaults 1 1
/dev/vg0/lv2 /recordfile2 xfs defaults 1 1
/dev/vg0/lv3 /recordfile3 xfs defaults 1 1
/dev/vg0/lv4 /recordfile4 xfs defaults 1 1
/dev/vg0/lv5 /recordfile5 xfs defaults 1 1
```

也可以在 rc 脚步中末尾添加 /sbin/mount －a 自动 mount 所有磁盘文件系统

(10) 测试数据读写操作。

从主机拷贝数据到 lv，查看是否有 IO 错误

从 lv 拷贝数据到主机，查看是否有 IO 错误

第6章 LVM 的其它功能

6.1 使用 snapshot 做备份

例 如我们要对卷组 "test_vg" 每晚进行数据库备份，就要采用 snapshot 类型的卷组。这种卷组是其它卷组的一个只读拷贝，它含有在创建 snapshot 卷组时原卷组的所有数据， 这意味你可以备份这个卷组而不用担心在备份过程中数据会改变，也不需要暂时关闭数据库卷以备份。

目标： 在安装后创建逻辑卷和阵列的技巧

6.1.1 使用 LVM 创建逻辑卷

(1) 使用 fdisk 在未分区空间创建四个新分区 ,类型为 Linux LVM (0x8e), 尺寸一样 ,为了加快速度 ,不要大于 1G. 退出时使用 w 保存更改 .不要重启动 .

(2) 编辑 /etc/modules.conf 中包含以下行 (RHEL 可以不用做以下修改):

alias block-major-58 lvm-mod

alias char-major-109 lvm-mod

使用当前内核创建 initrd

mkinintrd – f – v /boot/initrd-$(uname – r).img $(uname – r)

这个命令将使系统在启动时加载 lvm-mod 模块 ,启用 LVM

(3) 重启动系统

(4) 用 root 登录 , 运行 vgscan 初始化 LVM 配置文件

(5) 使用 pvcreate 将 LVM 分区初始化为物理卷 .假设分区为

/dev/hda9

/dev/hda10

/dev/hda11

/dev/hda12

命令为 ：pvcreate /dev/hda9 /dev/hda10 /dev/hda11 /dev/hda12

可以使用 pddisplay 查看分区信息

(6) 然后创建卷组 test0. 使用默认 4MB 的扩展尺寸 ,只包含一个物理卷

vgcreate test0 /dev/hda9

可以使用 pddisplay 查看信息

(7) 创建一个小逻辑卷 ,不要占用所有空间 . 使用 vgdisplay 的 VG size 和 PE/size 信息 ,比如创建一个 40M 的逻辑卷 :

lvcreate －L 40M －n data test0

可以使用 lvdisplay /dev/test0/data 确认命令执行了 .

(8) 在逻辑卷上创建 ext3 文件系统 : mke2fs －j /dev/test0/data

(9) 创建 /data 目录 . mount /dev/test0/data /data

(10) 复制文件到 /data. 可以创建一个大文件 : dd if= /dev/zero of=/data/bigfile bs=1024 count=20000

使用 df 检查 /data 的磁盘使用情况和剩余空间 . 确认能够正常使用 .可以编辑 /etc/fstab 来自动加载 /data. 重启动测试

6.1.2 使用逻辑卷

(1) 首先 , 卸载 /data. 使用 e2fsadm 扩展分区尺寸 : e2fsadm －L+50M /dev/test0/data

(2) 重加载 /dev/test0/data 到 /data, 确认文件 . 运行 df 检查 /data 的磁盘使用情况和剩余空间 .

(3) 使用剩余扩展创建第二个逻辑分区 . 运行 vgdisplay 查看 PE /size, 格式类似于 166/644MB, 这表示卷组包含 166 个扩展 ,664MB 剩余空间 . 创建一个占用 166 个扩展逻辑卷 /dev/test0/scratch, 命令为 :

lvcreate －l 166 －n scratch test0

(4) 格式化新卷 : mke2fs －j /dev/test0/scratch

(5) 把未使用的物理卷加入卷组 vgextend test0 /dev/hda10

(6) 如果再次运行 vgdisplay, 可以看到增加的扩展 .用 20MB 的扩展定义新逻辑卷 .

e2fsadm －L+20M /dev/test0/scratch

使用 lvdisplay 和 vgdisplay 确认成功

(7) 接下来用 /data 的只读快照创建新的逻辑卷 . 首先用只读选项加载 /data

mount －o remount,ro /data

(8) 快照不需要和父卷尺寸一致 ,我们假设不需要保存太多数据 , 可以设置为 5M

lvcreate －s －L 5M －n snap /dev/test0/data

(9) 现在重加载 /data 为读写状态

mount －o remount,rw /data

(10) 创建新加载点 /snap, 使用 mount /dev/test0/snap /snap 比较 /data 和/snap, 两者内容应该一致

(11) 运行命令 for I in$(seq 1 10); do echo $1 > /data/$1; done 将在 /data 下创建十个文件 ,名称从 1 到 10. 这个命令不影响 /snap, 可以用 lvdisplay /dev/test0/snap 检查

(12) 当快照逻辑卷不能容纳改变的块时 ,将被 LVM 自动删除 ,即使当前在加载状态 .(避免这一情况的方法是尺寸和父卷一致 ,或者及时用 lvextend 扩展尺寸)可以通过以下方式看到这一现象 :

rm /data/bigfile

for i in $(seq 1 10000); do echo $1 > /data/$1; done

(13) 在 /var/log/messages 里可以看到类似信息 :

Mar 19 16:30:02 station12 kernel: lvm --giving up to snapshot /dev/test0/data on /dev/test0/snap: out of space

运行 ls /snap. 快照已经不可用了 ,目录是空的 . 运行 lvdisplay /dev/test0/snap, 和 11 步的结果比较 .

(14) 做完快照之后 ,如果数据已经备份 ,或者快照已被删除 ,都需要被卸载 ,否则会造成轻微的性能下降 , 使用 umount /snap; lvremove /dev/test0/snap

在进行阵列试验以前清除 LVM 卷 :

删除所有 /etc/fstab 中增加的记录

umount /dev/test0/data; umount /dev/test0/scratch

lvremove /dev/test0/data; lvremove /dev/test0/scratch

vgchange －an test0; vgremove test0

6.2 用快照执行备份

如果在备份过程期间数据没有发生变化，那么就能够获得一致的备份。如果不在备份期间停止系统，就很难保证数据没有变化。

Linux LVM 实现了一种称为 快照（Snapshot ）的特性，它的作用就像是 “拍摄 ”逻辑卷在某一时刻的照片。 通过使用快照， 可以获得同一 LV 的两个拷贝 —— 一个可以用于备份，另一个继续用于日常操作。

快照有两大优点：

(1) 快照的创建非常快，不需要停止生产环境。

(2) 建立两个拷贝，但是它们的大小并不一样。快照使用的空间仅仅是存储两个 LV 之间的差异所需的空间。

快照由一个 例外列表（ exception list ）来实现，每当 LV 之间出现差异时就会更新这个列表（正式的说法是 CoW ， Copy-on-Write ）。

创建新的快照

创建新的快照 LV 也是使用 lvcreate 命令，但是要指定 -s 参数和原来的 LV 。在这种情况下， -L size 指定例外列表的大小，这影响快照支持的最大差异量，如果差异超过这个量，就无法保持一致性。

6.3 LVM2 系统管理功能

最后， 我要介绍一些可以用 LVM2 执行的系统管理任务，包括按需虚拟化、用镜像提高容错能力以及透明地对块设备执行加密。

快照和虚拟化

在使用 LVM2 时，快照可以不是只读的。这意味着，在创建快照之后，可以像常规块设备一样挂载和读写快照。

因为流行的虚拟化系统（比如 Xen 、VMWare 、 Qemu 和 KVM ）可以将块设备用作 guest 映像，所以可以创建这些映像的完整拷贝，并根据需要使用它们，它们就像是内存占用量很低的虚拟机。这样做的好处是部署迅速（创建快照的时间常常不超过几秒）和节省空间（ guest 共享原映像的大多数数据）。

设置的步骤如下：

(1) 为原映像创建一个逻辑卷。

(2) 使用这个 LV 作为磁盘映像安装 guest 虚拟机。

(3) 暂停这个虚拟机。内存映像可以是一个常规文件，所有其他快照都放在里面。

(4) 为原 LV 创建一个可读写的快照。

(5) 使用快照卷作为磁盘映像生成一个新的虚拟机。如果需要的话，要修改网络 /控制台设置。

(6) 登录已经创建的虚拟机，修改网络设置 /主机名。

完成这些步骤之后， 就可以让用户访问刚创建的虚拟机了。如果需要另一个虚拟机，那么只需重复步骤 4 到 6（所以不需要重新安装虚拟机）。还可以用一个脚本自动执行这些步骤。

在使用完虚拟机之后， 可以停止虚拟机并销毁快照。

更好的容错能力

最近的 LVM2 开发成果为逻辑卷提供了高可用性。逻辑卷可以有两个或更多的镜像，镜像可以放在不同的物理卷（或不同的设备）上。当在设备上发现 I/O 错误时，可以使用 dmeventd 让一个 PV 离线，而不会影响服务。更多信息请参考 lvcreate(8) 、lvconvert(8) 和 lvchange(8) 手册页。

如果硬件能够支持的话，可以用 dm_multipath 通过不同的通道访问同一设备，这样的话在一个通道发生故障时，可以转移到另一个通道。更多细节请参考 dm_multipath 和 multipathd 的文档。

透明的设备加密

可以用 dm_crypt 对块设备或逻辑卷执行透明的加密。 更多信息请参考 dm_crypt 的文档和 cryptsetup(8) 手册页。