LVM卷管理操作指导书

**关键词**

LVM 逻辑卷 pv vg lv

**描述:**

LVM是逻辑盘卷管理（LogicalVolumeManager)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（volumegroup ),形成一个存储池。管理员可以在卷组上随意创建逻辑卷组（logicalvolumes),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** | **LogicalVolumeManager( 逻辑卷管理)** |
| **PV** | **物理卷（physicalvolume )** |
| **VG** | **卷组（VolumeGroup )** |
| **LV** | **逻辑卷（logicalvolume )** |

**参考手册:**

LVMHowTo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 5](#_Toc18336688)

[1.1 前言 5](#_Toc18336689)

[1.2 什么是LVM 5](#_Toc18336690)

[1.3 为什么使用LVM 6](#_Toc18336691)

[1.3.1 小系统使用LVM的益处 6](#_Toc18336692)

[1.3.2 大系统使用LVM的益处 6](#_Toc18336693)

[第2章 LVM构成 7](#_Toc18336694)

[2.1 LVM的结构简图 7](#_Toc18336695)

[2.2 LVM的组成 7](#_Toc18336696)

[2.3 LVM的安装 8](#_Toc18336697)

[第3章 LVM的一般操作 8](#_Toc18336698)

[3.1 建立PV 8](#_Toc18336699)

[3.2 建立VG 9](#_Toc18336700)

[3.3 激活VG 9](#_Toc18336701)

[3.4 移除VG 10](#_Toc18336702)

[3.5 为VG增加新PV 10](#_Toc18336703)

[3.6 从VG移除 PV 10](#_Toc18336704)

[3.7 创建LV 11](#_Toc18336705)

[3.8 删除LV 12](#_Toc18336706)

[3.9 扩展 LV 12](#_Toc18336707)

[3.10 缩小LV 13](#_Toc18336708)

[3.11 在PV间转移数据 13](#_Toc18336709)

[3.12 系统启动/关闭LVM 13](#_Toc18336710)

[第4章 磁盘分区问题 14](#_Toc18336711)

[第5章 建立LVM示例1 14](#_Toc18336712)

[5.1 修改分区格式为 lvm 分区 14](#_Toc18336713)

[5.2 创建PV 15](#_Toc18336714)

[5.3 创建卷组 15](#_Toc18336715)

[5.4 建立 LV 16](#_Toc18336716)

[5.5 建立文件系统 17](#_Toc18336717)

[5.6 测试文件系统 17](#_Toc18336718)

[第6章 建立LVM示例2 17](#_Toc18336719)

[第7章 LVM 的其它功能 20](#_Toc18336720)

[7.1 使用 snapshot 做备份 20](#_Toc18336721)

[7.1.1 使用LVM创建逻辑卷 20](#_Toc18336722)

[7.1.2 使用逻辑卷 21](#_Toc18336723)

[7.2 快照备份优缺点 23](#_Toc18336724)

[7.3 LVM2 系统管理功能 24](#_Toc18336725)

# 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PartationMagic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,LogicalVolumeManager)机制就是一个完美的解决方案。

## 1.2 什么是LVM

LVM是LogicalVolumeManager(逻辑卷管理)的简写,它由HeinzMauelshagen在Linux2.4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（volumegroup),形成一个存储池。管理员可以在卷组上随意创建逻辑卷组（logicalvolumes),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

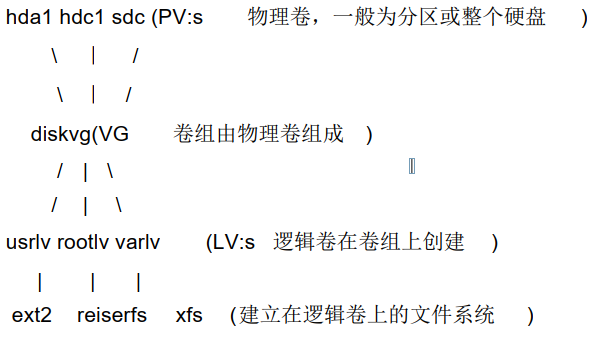
在使用很多硬盘的大系统中,使用LVM主要是方便管理、增加了系统的扩展性。

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volumegroup(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。

1. 物理卷physicalvolume(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SoftwareRAID卷。

1. 逻辑卷logicalvolume(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。

1. 物理块physicalextent(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。

1. 逻辑块logicalextent(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。

1. 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LE1–99映射到PV1,LE100–347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LE1映射为PV1的PE1,LE2映射为PV2的PE1,LE3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。

## LVM的安装

首先确定系统中是否安装了lvm 工具:

# rpm –qa|grep lvm2

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。

# 第3章 LVM的一般操作

## 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 /dev/mapper/mpath1 [ /dev/mapper/mpath2 ... ]

它的参数可以是整个磁盘、分区,也可以是loop设备。

## 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。

## 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。

## 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

## 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。

## 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

## 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another testlv 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。

## 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

## 扩展LV

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。

## 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。**xfs分区是不支持减小操作的.**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

## 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。**

## 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

LVM允许PV建立在几乎所有块设备上,如整个硬盘、硬盘分区

# 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。

# 第5章 建立LVM示例1

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 创建PV

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VG 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ext3系统:

# mkfs.ext3 /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。

# 第6章 建立LVM示例2

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。配置步骤如下:

(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.

1. 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7章 LVM 的其它功能

## 7.1 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

### 7.1.1 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)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pvdisplay查看分区信息

(3) 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vgdisplay查看信息

(4) 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.

(5) 在逻辑卷上创建ext3文件系统 :

mkfs.ext3 -j /dev/test0/data

1. 创建/data目录.

mount /dev/test0/data /data

(7) 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.

### 7.1.2 使用逻辑卷

(1) 首先卸载/data.使用 lvextend 扩展分区尺寸 :

lvextend –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

1. 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.

lvextend –L+20M /dev/test0/scratch

使用lvdisplay和vgdisplay确认成功

1. 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

(7) 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

(8) 现在重加载/data为读写状态

mount – o remount,rw /data

(9) 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

(10)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

(11)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

## 7.2 快照备份优缺点

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:

* 快照的创建非常快,不需要停止生产环境
* 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。

## 7.3 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:

为原映像创建一个逻辑卷。

使用这个LV作为磁盘映像安装guest虚拟机。

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。

为原 LV 创建一个可读写的快照。

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。

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.