RHEL ENTERPRISE 6.4 多路径软件 multi-path 配置操作手册

目录

一、 什么是多路径

普通的电脑主机都是一个硬盘挂接到一个总线上，这里是一对一的关系。而到了有光纤组成的 SAN 环境，或者由 iSCSI 组成的 IPSAN 环境，由于主机和存储通过了光纤交换机或者多块网卡及 IP 来连接，这样的话，就构成了多对多的关系。

也就是说，主机到存储可以有多条路径可以选择。主机到存储之间的 IO 由多条路径可以选择。每个主机到所对应的存储可以经过几条不同的路径，如果是同时使用的话，I/O 流量如何分配？其中一条路径坏掉了，如何处理？还有在操作系统的角度来看，每条路径，操作系统会认为是一个实际存在的物理盘，但实际上只是通向同一个物理盘的不同路径而已，这样是在使用的时候，就给用户带来了困惑。多路径软件就是为了解决上面的问题应运而生的。

另外在 linux 中，同样的设备在重新插拔、系统重启等情况下，自动分配的设备名称并非总是一致的，它们依赖于启动时内核加载模块的顺序，就有可能导致设备名分配不一致。

1.1 多路径的主要功能

多路径的主要功能就是和存储设备一起配合实现如下功能：

1.故障的切换和恢复

2.IO 流量的负载均衡

3.磁盘的虚拟化

由于多路径软件是需要和存储在一起配合使用的，不同的厂商基于不同的操作系统，都提供了不同的版本。并且有的厂商，软件和硬件也不是一起卖的，如果要使用多路径软件的话，可能还需要向厂商购买 license 才行。

比如 EMC 公司基于 linux 下的多路径软件，就需要单独的购买 license。好在， RedHat 和 Suse 的 2.6 的内核中都自带了免费的多路径软件包，并且可以免费使用，同时也是一个比较通用的包，可以支持大多数存储厂商的设备，即使是一些不是出名的厂商，通过对配置文件进行稍作修改，也是可以支持并运行的很好的。

1.2 UUID 的作用及意义

原因 1：它是真正的唯一标志符

UUID 为系统中的存储设备提供唯一的标识字符串，不管这个设备是什么类型的。如果你在系统中添加了新的存储设备如硬盘，很可能会造成一些麻烦，比如说启动的时候因为找不到设备而失败，而使用 UUID 则不会有这样的问题。

原因 2：设备名并非总是不变的

自动分配的设备名称并非总是一致的，它们依赖于启动时内核加载模块的顺序。如果你在插入了 USB 盘时启动了系统，而下次启动时又把它拔掉了，就有可能导致设备名分配不一致。如何让它保持在任何系统中的标识，那就是 UUID 唯一性标识。

二、 Linux 下 multipath 介绍

2.1 查看 multipath 是否安装

查看 multipath 是否安装如下：

[root@testvm1 disk]# rpm -qa |grep device-mapper

device-mapper-event-libs-1.02.74-10.el6.x86_64

device-mapper-multipath-libs-0.4.9-56.el6.x86_64

device-mapper-event-1.02.74-10.el6.x86_64

device-mapper-1.02.74-10.el6.x86_64

device-mapper-libs-1.02.74-10.el6.x86_64

device-mapper-multipath-0.4.9-56.el6.x86_64

2.2 Linux 下 multipath 需要以下工具包介绍

1、device-mapper-multipath：即 multipath-tools。

主要提供 multipathd 和 multipath 等工具和 multipath.conf 等配置文件。这些工具通过 device mapper 的 ioctr 的接口创建和配置 multipath 设备（调用 device-mapper 的用户空间库。创建的多路径设备会在 /dev/mapper 中）。

2、 device-mapper：

主要包括两大部分：内核部分和用户部分。

内核部分主要有 device mapper 核心（dm.ko）和一些 target driver（md-multipath.ko）。

核心完成设备的映射，而 target 根据映射关系和自身特点具体处理从 mappered device 下来的 i/o。同时，在核心部分，提供了一个接口，用户通过 ioctr 可和内核部分通信，以指导内核驱动的行为，比如如何创建 mappered device，这些 divece 的属性等。

用户空间部分主要包括 device-mapper 这个包。其中包括 dmsetup 工具和一些帮助创建和配置 mappered device 的库。这些库主要抽象、封装了与 ioctr 通信的接口，以便方便创建和配置 mappered device。multipath-tool 的程序中就需要调用这些库。

3、dm-multipath.ko 和 dm.ko：

dm.ko 是 device mapper 驱动。它是实现 multipath 的基础。dm-multipath 其实是 dm 的一个 target 驱动。

4、scsi_id：

包含在 udev 程序包中，可以在 multipath.conf 中配置该程序来获取 scsi 设备的序号。通过序号，便可以判断多个路径对应了同一设备。这个是多路径实现的关键。

scsi_id 是通过 sg 驱动，向设备发送 EVPD page80 或 page83 的 inquery 命令来查询 scsi 设备的标识。但一些设备并不支持 EVPD 的 inquery 命令，所以他们无法被用来生成 multipath 设备。但可以改写 scsi_id，为不能提供 scsi 设备标识的设备虚拟一个标识符，并输出到标准输出。

multipath 程序在创建 multipath 设备时，会调用 scsi_id，从其标准输出中获得该设备的 scsi id。在改写时，需要修改 scsi_id 程序的返回值为 0。因为在 multipath 程序中，会检查该直来确定 scsi id 是否已经成功得到。

三、 multipath 在 Redhat 中的基本配置过程

3.1 安装和加载多路径软件包

```
# rpm -ivh device-mapper-1.02.39-1.el5.rpm    #安装映射包
```

rpm -ivh device-mapper-multipath-0.4.7-34.el5.rpm #安装多路径包

或者使用 yum 进行安装

yum -y install device-mapper-multipath-libs.x86_64
yum -y isntall device-mapper-multipath.x86_64

3.2 设置开机启动

#是否开机自启动
[root@testvm1 dev]# chkconfig --list|grep multipathd
multipathd 0:off 1:off 2:off 3:off 4:off 5:off 6:off

chkconfig --level 2345 multipathd on #设置成开机自启动 multipathd

modprobe -l |grep multipath #来检查安装是否正常,内核中是否存在
kernel/drivers/md/dm-multipath.ko

至此进行了安装并设置了开机启动，但是 multipath 服务现在还没有启动，如果启动两种办法：
一、重启启动系统 reboot ，开机自动载入内核并启动服务。
二、手工加载

modprobe dm-multipath #加载到内核
#modprobe dm-round-robin
service multipathd start #重启服务

3.3 生成 multipath 配置文件

用/sbin/mpatchconf --enable 生成 multipath.conf
[root@testvm2 ~]# /sbin/mpatchconf --enable
[root@testvm2 ~]# ls /etc/multipath.conf
/etc/multipath.conf
(也可以按上面提示将/usr/share/doc/device-mapper-multipath-0.4.9/multipath.conf 文件复制到/etc 下)

四、 multipath 高级配置

除了可以 multipath 命令来的默认配置 multipath，比如映射设备的名称、multipath 负载均衡。

也可以按照我们自己定义的方法来配置 multipath。
首先，需要获取 uuid

4.1 获取存储设备的 UUID/wwid 和路径

通过/sbin/scsi_id -g -u -s /block/sdf 获取 uuid/wwid
通过 multipath -v3 命令查看，注意，会默认生成设备的路径。
multipath -v3

```
===== paths list =====
uuid                                hcil     dev dev_t pri dm_st chk_st vend/prod
                                    2:0:0:0 sda 8:0   1   undef ready  VMware,,V
36000c29edb0c86955aea188e77bdd6f1 3:0:0:0 sdb 8:16  1   undef ready  VMware,,V
36000c298dc652acd46c792e06b2d4198 3:0:1:0 sdc 8:32  1   undef ready  VMware,,V
Jun 12 18:10:34 | params = 0 0 1 1 round-robin 0 1 1 8:32 1
Jun 12 18:10:34 | status = 2 0 0 0 1 1 A 0 1 0 8:32 A 0
Jun 12 18:10:34 | params = 0 0 1 1 round-robin 0 1 1 8:16 1
Jun 12 18:10:34 | status = 2 0 0 0 1 1 A 0 1 0 8:16 A 0
```

1、通过命令查看：
例如：/sbin/blkid
/sbin/blkid /dev/sdg1
但是只能看到已挂接文件系统的存储和分区的 uuid，对于裸设备、未挂接的分区看不到 uuid。
2、文件查看：ls -l /dev/disk/by-uuid
3、查看文件 ls -l /dev/disk/by-id

```
[root@testvm1 mapper]# ls -l /dev/disk/by-id/
total 0
lrwxrwxrwx 1 root root 10 Jun 12 18:00 dm-name-mpatha -> ../../dm-2
lrwxrwxrwx 1 root root 10 Jun 12 18:00 dm-name-mpathap1 -> ../../dm-4
lrwxrwxrwx 1 root root 10 Jun 12 18:00 dm-name-mpathap2 -> ../../dm-5
lrwxrwxrwx 1 root root 10 Jun 12 18:00 dm-name-mpathap3 -> ../../dm-6
lrwxrwxrwx 1 root root 10 Jun 12 18:00 dm-name-mpathb -> ../../dm-3
lrwxrwxrwx 1 root root 10 Jun 12 13:39 dm-name-vg_testvm1-lv_root -> ../../dm-0
lrwxrwxrwx 1 root root 10 Jun 12 13:39 dm-name-vg_testvm1-lv_swap -> ../../dm-1
lrwxrwxrwx 1 root root 10 Jun 12 13:39 dm-uuid-LVM-zHKHkRpzSGLxdPMDFOQk6X8es3Eb9srlImGnT4f9ErA5dtI7adcfThOLG5DhmCer -> ../../dm-0
lrwxrwxrwx 1 root root 10 Jun 12 13:39 dm-uuid-LVM-zHKHkRpzSGLxdPMDFOQk6X8es3Eb9srlQKmIK83hrtgFKF6F6RPgx2Tro2A052kZ -> ../../dm-1
lrwxrwxrwx 1 root root 10 Jun 12 18:00 dm-uuid-mpath-36000c298dc652acd46c792e06b2d4198 -> ../../dm-3
lrwxrwxrwx 1 root root 10 Jun 12 18:00 dm-uuid-mpath-36000c29edb0c86955aea188e77bdd6f1 -> ../../dm-2
lrwxrwxrwx 1 root root 10 Jun 12 18:00 dm-uuid-part1-mpath-36000c29edb0c86955aea188e77bdd6f1 -> ../../dm-4
lrwxrwxrwx 1 root root 10 Jun 12 18:00 dm-uuid-part2-mpath-36000c29edb0c86955aea188e77bdd6f1 -> ../../dm-5
lrwxrwxrwx 1 root root 10 Jun 12 18:00 dm-uuid-part3-mpath-36000c29edb0c86955aea188e77bdd6f1 -> ../../dm-6
lrwxrwxrwx 1 root root  9 Jun 12 17:31 scsi-36000c298dc652acd46c792e06b2d4198 -> ../../sdc
lrwxrwxrwx 1 root root  9 Jun 12 17:31 scsi-36000c29edb0c86955aea188e77bdd6f1 -> ../../sdb
lrwxrwxrwx 1 root root 10 Jun 12 17:31 scsi-36000c29edb0c86955aea188e77bdd6f1-part1 -> ../../sdb1
lrwxrwxrwx 1 root root 10 Jun 12 17:31 scsi-36000c29edb0c86955aea188e77bdd6f1-part2 -> ../../sdb2
lrwxrwxrwx 1 root root 10 Jun 12 17:31 scsi-36000c29edb0c86955aea188e77bdd6f1-part3 -> ../../sdb3
lrwxrwxrwx 1 root root  9 Jun 12 17:31 wwn-0x6000c298dc652acd46c792e06b2d4198 -> ../../sdc
lrwxrwxrwx 1 root root  9 Jun 12 17:31 wwn-0x6000c29edb0c86955aea188e77bdd6f1 -> ../../sdb
lrwxrwxrwx 1 root root 10 Jun 12 17:31 wwn-0x6000c29edb0c86955aea188e77bdd6f1-part1 -> ../../sdb1
lrwxrwxrwx 1 root root 10 Jun 12 17:31 wwn-0x6000c29edb0c86955aea188e77bdd6f1-part2 -> ../../sdb2
lrwxrwxrwx 1 root root 10 Jun 12 17:31 wwn-0x6000c29edb0c86955aea188e77bdd6f1-part3 -> ../../sdb3
```

其中红线部分既是 uuid。

看路径 ls -l /dev/disk/by-path/

```
[root@testvm1 mapper]# ls -l /dev/disk/by-path/
total 0
lrwxrwxrwx 1 root root  9 Jun 12 17:31 pci-0000:00:10.0-scsi-0:0:0:0 -> ../../sda
lrwxrwxrwx 1 root root 10 Jun 12 13:39 pci-0000:00:10.0-scsi-0:0:0:0-part1 -> ../../sda1
lrwxrwxrwx 1 root root 10 Jun 12 13:39 pci-0000:00:10.0-scsi-0:0:0:0-part2 -> ../../sda2
lrwxrwxrwx 1 root root  9 Jun 12 13:39 pci-0000:02:07.0-scsi-1:0:0:0 -> ../../sr0
lrwxrwxrwx 1 root root  9 Jun 12 17:31 pci-0000:02:08.0-scsi-0:0:0:0 -> ../../sdb
lrwxrwxrwx 1 root root 10 Jun 12 17:31 pci-0000:02:08.0-scsi-0:0:0:0-part1 -> ../../sdb1
lrwxrwxrwx 1 root root 10 Jun 12 17:31 pci-0000:02:08.0-scsi-0:0:0:0-part2 -> ../../sdb2
lrwxrwxrwx 1 root root 10 Jun 12 17:31 pci-0000:02:08.0-scsi-0:0:0:0-part3 -> ../../sdb3
lrwxrwxrwx 1 root root  9 Jun 12 17:31 pci-0000:02:08.0-scsi-0:0:1:0 -> ../../sdc
```

4.2 配置/etc/multipath.conf 文件例子

1、 查看设备

ls -l /dev/

```
brw-rw----  1 root disk      8,  16 Jun 12 18:39 sdb
brw-rw----  1 root disk      8,  32 Jun 12 17:31 sdc
```

2、 获取设备的 uuid

使用 multipath 设备名生成设备路径，同时也获取了设备 uuid

```
[root@testvm1 mapper]# multipath /dev/sdb
create: mpatha (36000c29edb0c86955aea188e77bdd6f1) undef VMware,,VMware Virtual S
size=3.0G features='0' hwhandler='0' wp=undef
`-+- policy='round-robin 0' prio=1 status=undef
  `- 3:0:0:0 sdb 8:16 undef ready running
[root@testvm1 mapper]# multipath /dev/sdc
create: mpathb (36000c298dc652acd46c792e06b2d4198) undef VMware,,VMware Virtual S
size=3.0G features='0' hwhandler='0' wp=undef
`-+- policy='round-robin 0' prio=1 status=undef
  `- 3:0:1:0 sdc 8:32 undef ready running
```

3、 配置/etc/multipath.conf 文件

vi /etc/multipath.conf

```
#multipaths {
#       multipath {
#               wwid                    3600508b4000156d700012000000b0000
#               alias                   yellow
#               path_grouping_policy    multibus
#               path_checker            readsector0
#               path_selector           "round-robin 0"
#               failback                manual
#               rr_weight               priorities
#               no_path_retry           5
#       }
#       multipath {
#               wwid                    1DEC_____321816758474
#               alias                   red
#       }
#}
```

将这段中前面的#号删除，或复制后编辑成实际需要的路径配置。

```
multipaths {
        multipath {
                wwid                    36000c29edb0c86955aea188e77bdd6f1
                alias                   mpath0
                path_grouping_policy    multibus
#               path_checker             readsector0
                path_selector           "round-robin 0"
                failback                manual
                rr_weight               priorities
                no_path_retry           5
        }
        multipath {
                wwid                    36000c298dc652acd46c792e06b2d4198
                alias                   mpath1
        }
}
```

配置了设备 sdb 、sdc 的多路径

运行 multipath 命令生成路径文件(如果之前已经有该设备的路径文件，不会重新生成)

```
[root@testvm1 mapper]# multipath
create: mpath0 (36000c29edb0c86955aea188e77bdd6f1) undef VMware,,VMware Virtual S
size=3.0G features='0' hwhandler='0' wp=undef
`-+- policy='round-robin 0' prio=1 status=undef
  `- 3:0:0:0 sdb 8:16 undef ready running
create: mpath1 (36000c298dc652acd46c792e06b2d4198) undef VMware,,VMware Virtual S
size=3.0G features='0' hwhandler='0' wp=undef
`-+- policy='round-robin 0' prio=1 status=undef
  `- 3:0:1:0 sdc 8:32 undef ready running
[root@testvm1 mapper]# ll
total 0
crw-rw---- 1 root root 10, 58 Jun 12 13:39 control
lrwxrwxrwx 1 root root       7 Jun 12 19:45 mpath0 -> ../dm-2
lrwxrwxrwx 1 root root       7 Jun 12 19:45 mpath1 -> ../dm-3
lrwxrwxrwx 1 root root       7 Jun 12 13:39 vg_testvm1-lv_root -> ../dm-0
lrwxrwxrwx 1 root root       7 Jun 12 13:39 vg_testvm1-lv_swap -> ../dm-1
```

查看

multipath -ll

```
[root@testvm1 mapper]# multipath -ll
mpath1 (36000c298dc652acd46c792e06b2d4198) dm-3 VMware,,VMware Virtual S
size=3.0G features='0' hwhandler='0' wp=rw
`-+- policy='round-robin 0' prio=1 status=active
  `- 3:0:1:0 sdc 8:32 active ready running
mpath0 (36000c29edb0c86955aea188e77bdd6f1) dm-2 VMware,,VMware Virtual S
size=3.0G features='1 queue_if_no_path' hwhandler='0' wp=rw
`-+- policy='round-robin 0' prio=1 status=active
  `- 3:0:0:0 sdb 8:16 active ready running
```

如果是通过光纤多条线路连接的会显示多条连接线路复合成一条链路，这个只是本地硬盘所以只有一条路径，类似下面的信息：

```
mpath0 (360060e80058e980000008e9800000007)
[size=20 GB][features="0"][hwhandler="0"]
\_ round-robin 0 [prio=1][active]
\_ 3:0:0:7 sdaa 65:160 [active][ready]
\_ round-robin 0 [prio=1][enabled]
\_ 4:0:0:7 sdas 66:192 [active][ready]
\_ round-robin 0 [prio=1][enabled]
\_ 5:0:0:7 sdbk 67:224 [active][ready]
\_ round-robin 0 [prio=1][enabled]
\_ 2:0:0:7 sdi 8:128 [active][ready]
```

如果针对设备路径 mpath0 进行分区，

fdisk /dev/mapper/mpath0

```
Disk /dev/mapper/mpath0: 3221 MB, 3221225472 bytes
255 heads, 63 sectors/track, 391 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x83ee84ba

                Device Boot      Start         End      Blocks   Id  System
/dev/mapper/mpath0p1               1          65      522081   83  Linux
/dev/mapper/mpath0p2              66         168      827347+  83  Linux
/dev/mapper/mpath0p3             169         194      208845   83  Linux
/dev/mapper/mpath0p4             195         391     1582402+  83  Linux
```

fdisk 对多路径软件生成的磁盘进行分区之后，所生成的磁盘分区并没有马上添加到/dev/目录下，

此时我们要重启 IPSAN 或者 FCSAN 的驱动.

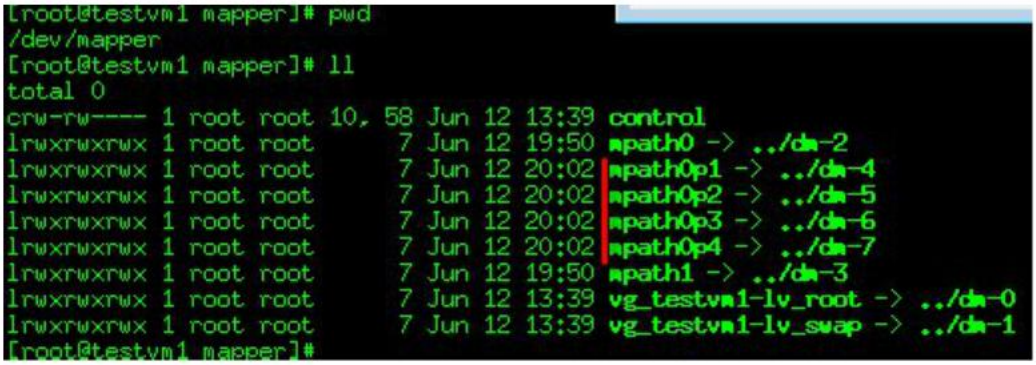
如果是用 iscsi-initiator 来连接 IPSAN 的重启 ISCSI 服务就可以发现所生成的磁盘分区了

service iscsi restart

如果是本地磁盘可以使用 partprobe 同步磁盘信息(分完区使用 partprobe　同步磁盘信息(此命令让 kernel 会重新读取磁盘分区表，修改生效)或 partprobe　/dev/mapper/mpath0　单独同步)

partprobe /dev/mapper/mpath0

在/dev/mapper 下已有了分区的路径文件



4.3 关于：scsi_id

其包含在 udev 程序包中，可以在 multipath.conf 中配置该程序来获取 scsi 设备的序号。通过序号，便可以判断多个路径对应了同一设备。这个是多路径实现的关键。scsi_id 是通过 sg 驱动，向设备发送 EVPD page80 或 page83 的 inquery 命令来查询 scsi 设备的标识。但一些设备并不支持 EVPD 的 inquery 命令，所以他们无法被用来生成 multipath 设备。但可以改写 scsi_id，为不能提供 scsi 设备标识的设备虚拟一个标识符，并输出到标准输出。

multipath 程序在创建 multipath 设备时，会调用 scsi_id，从其标准输出中获得该设备的 scsi id。在改写时，需要修改 scsi_id 程序的返回值为 0。因为在 multipath 程序中，会检查该直来确定 scsi id 是否已经成功得到。

五、 multipath 基本命令

# multipath	自动生成设备路径，对于没有生成对于没有默认路径的磁盘设备或分区进行自动生成，不会重复生成，包括磁盘的，以及磁盘上的分区的. 如果有磁盘 sdb 、sdc，磁盘 sdb 分了四个区，运行 multipath
# multipath 设备名	仅对该设备生成路径 ,不能针对设备的分区，针对的是设备 命令将自动生成设备路径，在/dev/mapper 下会有 mpatha、mpathap1 、mpathap2、mpathap3、mpathap4、mpathb 等路径。其中 mpathap1 、mpathap2、mpathap3、mpathap4 是 mpatha 的分区。 实际生成中应先不进行分区，在磁盘的路径上再使用 fdisk 进行分区.
# multipath -v<1 234>	#格式化路径，检测路径，合并，自动生成设备路径，对于没有生成对于没有默认路径的磁盘设备或分区进行自动生成，不会重复生成，包括磁盘的，以及磁盘上的分区的. 数字 1 - 4，显示的信息不同。
# multipath -F	#删除现有没有使用的路径,将没有挂载的文件系统、绑定为 raw 设备的路径删除
# multipath -ll	#查看多路径状态,不论是 multipath.conf 文件中配置的还是未配置而使用 multipath 命令自动生成
#multipath ll	#查看多路径状态,只显示 multipath.conf 文件中配置的.

六、 multipath.conf 配置文件说明

multipath.conf 主要包括 blacklist、multipaths、devices 三部份的配置
blacklist 配置

```
blacklist {
        wwid 26353900f02796769
        devnode "^(ram|raw|loop|fd|md|dm-|sr|scd|st)[0-9]*"
        devnode "^hd[a-z]"
}
```

将所有设备加入黑名单

Multipaths 部分配置 multipaths

```
multipaths {
         multipath {
wwid                    3600508b4000156d700012000000b0000    #此值 multipath -v3 可以看到
alias                   yellow                               #映射后的别名,可以随便取
path_grouping_policy    multibus                             #路径组策略
path_checker            readsector0                          #决定路径状态的方法
path_selector           "round-robin 0"          #选择那条路径进行下一个 IO 操作的方法
failback                manual
rr_weight               priorities
no_path_retry           5
                }
        }
```

Devices 部分配置

```
devices {
        device {
vendor                  "COMPAQ   "              #厂商名称
product                 "HSV110 (C)COMPAQ"       #产品型号
path_grouping_policy    multibus                 #默认的路径组策略
getuid_callout          "/lib/udev/scsi_id --whitelisted --device=/dev/%n"   #获得唯一设备号使用的默认程序
prio_callout            "/sbin/acs_prio_alua %d"   #获取有限级数值使用的默认程序
path_checker            readsector0                #决定路径状态的方法
path_selector           "round-robin 0"            #选择那条路径进行下一个 IO 操作的方法
hardware_handler        "0"
failback                15                         #故障恢复的模式
rr_weight               priorities
no_path_retry           queue      #在 disable queue 之前系统尝试使用失效路径的次数的数值
rr_min_io               100              #在当前的用户组中，在切换到另外一条路径之前的 IO 请求的数目
        }
        device {
                vendor                  "COMPAQ   "
                product                 "MSA1000          "
```

```
        path_grouping_policy        multibus
    }
}
```

对一些设备名可以进行黑名单过滤，如：

```
blacklist {

devnode    "^(ram|raw|loop|fd|md|dm-|sr|scd|st)[0-9]*"

devnode    "^hd[a-z][[0-9]*]"

devnode     "^cciss!c[0-9]d[0-9]*
 devnode "^sda"
}
```

七、 对 multipath 磁盘的基本操作

要对多路径软件生成的磁盘进行操作直接操作/dev/mapper/目录下的磁盘就行.

对多路径软件生成的磁盘进行分区，用 fdisk 对多路径软件生成的磁盘进行分区保存时会有一个报错,此报错不用理会。

fdisk /dev/mapper/mpatha

在使用 fdisk -l 查看

```
[root@testvm1 mapper]# fdisk -l /dev/mapper/mpatha

Disk /dev/mapper/mpatha: 3221 MB, 3221225472 bytes
255 heads, 63 sectors/track, 391 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x83ee84ba

            Device Boot      Start         End      Blocks   Id  System
/dev/mapper/mpathap1               1          65      522081   83  Linux
/dev/mapper/mpathap2             102         204      827347+  83  Linux
/dev/mapper/mpathap3              66         101      289170   83  Linux
```

fdisk 对多路径软件生成的磁盘进行分区之后，所生成的磁盘分区并没有马上添加到/dev/目录下，

此时我们要重启 IPSAN 或者 FCSAN 的驱动.

如果是用 iscsi-initiator 来连接 IPSAN 的重启 ISCSI 服务就可以发现所生成的磁盘分区了

service iscsi restart

如果是本地磁盘可以使用 partprobe 同步磁盘信息

ls -l /dev/mapper/

对通过网络方式连接的存储，如果需要在开机自动挂接到文件系统的/etc/fstab 文件配置时，最好加上在 Defaults 前记得加上参数“_netdev”.
此参数的意思是说在网络服务启动完成后再执行 mount 操作,否则启动过程可能报错。如：

```
/dev/mapper/mpathap1                    /data1                          ext3       _netdev,defaults    0 0
```

对其中的一个分区进行格式化

```
[root@testvm1 mapper]# mkfs -t ext4 /dev/mapper/mpath0p1
mke2fs 1.41.12 (17-May-2010)
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
Stride=0 blocks, Stripe width=0 blocks
130560 inodes, 522080 blocks
26104 blocks (5.00%) reserved for the super user
First data block=1
Maximum filesystem blocks=67633152
64 block groups
8192 blocks per group, 8192 fragments per group
2040 inodes per group
Superblock backups stored on blocks:
        8193, 24577, 40961, 57345, 73729, 204801, 221185, 401409

Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 39 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
[root@testvm1 mapper]# mkfs -t ext4 /dev/mapper/mpath0p2
mke2fs 1.41.12 (17-May-2010)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
51744 inodes, 206836 blocks
10341 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=213909504
7 block groups
32768 blocks per group, 32768 fragments per group
7392 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840

Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 34 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
```

将两个分区挂接到文件系统

```
[root@testvm1 mapper]#  mount -t ext4 /dev/mapper/mpath0p1 /data1
[root@testvm1 mapper]#  mount -t ext4 /dev/mapper/mpath0p2 /data2
```

```
[root@testvm1 mapper]# vi /etc/fstab

#
# /etc/fstab
# Created by anaconda on Sat May 24 03:11:49 2014
#
# Accessible filesystems, by reference, are maintained under '/dev/disk'
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info
#
/dev/mapper/vg_testvm1-lv_root /                       ext4    defaults        1 1
UUID=879eab67-c427-40b3-b3ac-913dd4c0f2ae /boot                   ext4    defaults        1
/dev/mapper/vg_testvm1-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
/dev/mapper/mpath0p1    /data1                  ext4    _netdev,defaults  0 0
/dev/mapper/mpath0p2    /data2                  ext4    _netdev,defaults  0 0
```

将其他两个两个分区绑定到 raw。

```
[root@testvm2 data1]# /bin/raw /dev/raw/raw1  /dev/mapper/mpath0p3
/dev/raw/raw1:  bound to major 253, minor 6
[root@testvm2 data1]# /bin/raw /dev/raw/raw2  /dev/mapper/mpath0p4
/dev/raw/raw2:  bound to major 253, minor 7
[root@testvm2 data1]# ll /dev/raw
total 0
crw-rw---- 1 root disk 162, 1 Jun 12 20:31 raw1
crw-rw---- 1 root disk 162, 2 Jun 12 20:31 raw2
crw-rw---- 1 root disk 162, 0 Jun 12 19:43 rawctl
[root@testvm2 data1]# raw -qa
/dev/raw/raw1:  bound to major 253, minor 6
/dev/raw/raw2:  bound to major 253, minor 7
```

八、 使用 **multipath** 的一个例子

九、 **PV/VG/LV** 常用操作命令

常用命令：

pvcreate /dev/md0 #创建 PV

pvcreate /dev/mapper/mpath{1,2,3,4,5} 注意括号中的用法

pvscan 检索 pv

#pvdisplay 显示 pv 信息，

后跟 pv 名称可显示该 pv 详细信息

pvremove /dev/sdb1 删除物理卷

vgcreate LVM1 /dev/md0 #创建 VG

创建 vg，。一个 vg 中可以有一个 pv 也可以有多个 pv，如：

vgcreate vg1 /dev/mpath/mpath0 /dev/mpath/mpath1

vgcreate vg2 /dev/mpath/mpath2

vgcreate vg3 /dev/mpath/mpath3 /dev/mpath/mpath4 /dev/mpath/mpath5

vgscan　检索 vg

#vgdisplay　显示 vg 信息，

后跟 vg 名称可显示该 vg 详细信息

#vgextend　vgxx　pv 名称　将 vg 扩展 pv

为 VG 添加新的 PV 使其增加容量# vgextend vg1 /dev/hda7

删除 VG　# vgremove vg1

lvcreate -L 1.5TB -n data1 LVM1　#创建 LV　-L 指扩展的意思 创建一个 lv ，使用的 vg 如：

lvcreate -L 325GB -n data2 LVM1　#创建 LV

lvcreate -L 1G -n lv1 vg1

lvcreate -L 15G -n lv2 vg2

lvcreate -L 15G -n lv3 vg3

删除 LV# lvremove /dev/vg1/lv1

lvscan　#查看 LV 信息

#lvdisplay　显示 lv 信息，

后跟 lv 名称可显示该 lv 详细信息。

#lvremove　删除 LV（lvremove） 后跟 lv 名称

lvextend -L 1.5G /dev/vg1/lv1　例如将 1G 的/dev/vg1/lv1 修改成 1.5G：

说明：在这里-L　1.5G 意思是扩大“到”1.5G，而不是扩大 1.5G

当然，如果这个 lv 已经被格式化、被使用，这样还没有算完成，还要扩大文件系统，可以参考以下的办法：

resize2fs　/dev/vg1/lv1

lvextend.有两种方法，一个是指定在现有的空间上增加的大小，一个是指定将现有空间增加到多少。

在 testlv 现有空间的基础上再增加 10G

lvextend -L +10G -f -r /dev/testvg/testlv

#此时 testlv 的大小是 30G

将 testlv 的空间扩大到 100G

lvextend -L 100G -f -r /dev/testvg/testlv

#此时 testlv 的大小是 100G

减少 lv 大小：

不建议采取缩小 LV 的操作，如果非要缩小 LV，建议采用以下步骤：

a. 备份原 LV 上的数据（fbackup 或用其他软件，或 tar 到磁带机上或其他地方）

b. 删除原 LV（lvremove）

c. 创建新 LV（lvcreate）

d. 生成新的文件系统（newfs）

e. 恢复原 LV 上的数据（restore 或用其他软件，或解 tar 回来）

而如果有 OnlineJFS，可以采用以下办法：

首先检查文件系统是否有错 e2fsck -f　/dev/vg1/lv1

取消挂载 umount /dev/vg1/lv1

调整文件系统大小：　resize2fs　/dev/vg1/lv1　1G

调整 lv 的大小 : #lvreduce -L <new size in MB> /dev/vg01/lvdata

lvreduce -L 1G /dev/vg1/lv1

调整文件系统大小：　resize2fs　/dev/vg1/lv1

挂载

mount　-t　ext3 /dev/vg1/lv1 /data1

要配置 LVM，可以按以下步骤进行：

1. 创建和初始化物理卷（Physical Volume），通过 pvcreate 建立 pv，即 pv 阶段；
2. 添加物理卷到卷组（Volume Group），使用 vgcreate 加入多个 pv 成为 vg，即 vg 阶段；
3. 在卷组上创建逻辑卷（logical volume），使用 lvcreate 划分 vg,成为一个或多个 lv,即 lv 阶段；

针对创建的 lv 如果作为文件系统使用，还需要进行格式化，并挂接到文件系统上。
如，格式化：

```
mkfs.ext3 /dev/vg1/lv1
```

挂载到文件系统挂接点/data1

```
mount -t ext3    /dev/vg1/lv1 /data1
```

设置开机自动挂载：

编辑/etc/fstab

```
/dev/vg1/lv1 /data1 ext3        _netdev,defaults    0 0
```

如果需要修改 LV 的名字，则只要简单的做以下操作：

```
#umount /dev/vg01/lvol1
#mv /dev/vg01/lvol1 /dev/vg01/lvdata
#mv /dev/vg01/rlvol1 /dev/vg01/rlvdata
#mount /dev/vg01/lvdata <mountpoint>
```

修改 PV

有关 PV 的参数选项中，有一个是最常用的：-t，它是 LVM 对硬盘相应所等待的时间（timeout 值），默认的值是 30 秒。可以用一下命令修改成 120 秒的 timeout 值：

```
#pvchange -t 120 /dev/dsk/cXtXdX
```

如果要去掉 timeout，可以用以下命令：

```
#pvchange -t 0 /dev/dsk/cXtXdX
```

修改 VG

vgchange 命令可以用来激活/不激活 VG。

其中 max_pe 这个参数只能在 VG 创建的时候指定，默认的 PE 数是 1016。由于默认的 pe_size 的大小是 4M，而如果采用默认的 max_pe 的话，我们只能使用到 4G 的空间。这在动辄上百 G 的硬盘时代，这些默认值肯定不符合需求了。

除了在创建 PV 的时候指定 pe_size 更大以外（但这有个缺点，就是容易造成空间的浪费），还能够在创建 VG 的时候指定 max_pe，虽然默认值是 1016，但是实际上，LVM 会根据硬盘的实际大小和 pe_size 来决定 max_pe——不过这又引起另外一个问题，例如：我们现在使用的硬盘是 36G，而后来空间不够，我们又加了一个 72G 的硬盘，这个时候，由于在创建 VG 的时候，max_pe 已经固定了，这可能就会导致空间的浪费。我们可以根据数据的增量，来考虑设定 max_pe 的值。而 max_pe 的值，可以在 1——65535 之间。

为了修改 VG 的名字，我们可以有两个办法来实现：

（即系统默认仅 vg00 是加了 tag 的，其他 vg 要激活必须手动在某一节点上加 tag 来激活卷）

修改名字要保证 vg 是激活

vgchange --addtag tagname emcvg02（加 tag）

vgchange -ay emcvg02（激活卷）

改名过程：

改 lv 名

（此文件系统需 umount 状态）

lvchange -an /dev/newvg2/newlv2（去激活才能改名）

lvrename /dev/vg2/lv2 /dev/newvg2/newlv2

改 vg 名

vgrename emcvg02 newvgname

激活 newlv2

lvchange -ay /dev/newvg2/newlv2

注意：

如果使用 multipath 进行设备多路径绑定，有部分 multipath 版本存在与 lvm 兼容的问题。当使用 device-mapper 设备创建 lvm 完成，重启后，虽然 lvm 仍存在，但/dev/mapper 下的设备丢失。可以参考：

https://bugs.launchpad.net/ubuntu/+source/multipath-tools/+bug/230006

解决方法：

/etc/lvm/lvm.conf 文件中加入：
types=["device-mapper", 1]

十、 使用 udev 配置固定 iSCSI 磁盘设备名称

相同名称的设备文件在不同的系统中可能对应的是不同的磁盘。以下展示了一个实例，挂载到本地服务器的设备名称都是/dev/sdd，但对应的却不是同一个 iSCSI 磁盘。

节点 1 通过执行 fdisk -l 查看到的/dev/sdd 设备文件大小情况如下：

```
1. Disk /dev/sdd: 5502 MB, 5502926848 bytes
2. 170 heads, 62 sectors/track, 1019 cylinders
3. Units = cylinders of 10540 * 512 = 5396480 bytes
```

节点 1/dev/sdd 设备文件对应的 LUN 大小为 5502MB。

节点 2 通过执行 fdisk -l 查看到的/dev/sdd 设备文件大小情况如下：

```
1. Disk /dev/sdd: 1073 MB, 1073741824 bytes
2. 34 heads, 61 sectors/track, 1011 cylinders
3. Units = cylinders of 2074 * 512 = 1061888 bytes
```

节点 2/dev/sdd 设备文件对应的 LUN 大小为 1073MB。显然，两个节点/dev/sdd 对应的并不是同一个设备。

如果在环境中使用/dev/sd*的设备文件来管理和使用存储，而相同名称的设备文件对应的 iSCSI 磁盘却是不同的，这样就会导致操作的失败。例如，在第一个节点将分区/dev/sdd1 格式化成 OCFS2 挂载到某个节点，在第二个节点执行同样的挂载命令就会失败，因为第二个节点的/dev/sdd1 分区和第一个节点并不是同一个分区，即使挂载成功也不是相同的共享存储。

Linux 平台为了解决这个问题，使两个节点的设备文件都能相互对应，需要使用 udev 动态设备文件管理工具。它是一个默认安装并在系统启动时最先加载的工具，使用它能配置相应的设备加载规则，udev 通过定义的规则来生成相应的设备文件。在指定规则下能够使用磁盘唯一标识的属性作为生成设备文件名称的一部分，就能在不同的节点保证相同名称的设备文件指向相同的 iSCSI 磁盘。

下面举例说明 udev 的配置方法。

udev 的配置主目录是/etc/udev，包含以下文件和目录：

udev.conf 文件。这是 udev 输出日志的配置文件，默认的配置是：udev_log=“err”，udev_log 还可以被配置为 info 或 debug。一般默认即可，如果修改为 info 或 debug 将有大量的日志信息被输出。

rules.d 目录。此目录是最重要的配置目录，里面包含的都是配置的 udev 加载规则，udev 会根据配置的规则来生成相应的设备文件。所有的规则文件都必须以“.rules”作为扩展名。

scripts 目录。此目录保存着加载规则需要执行的脚本。

在/etc/udev/rules.d 目录下创建 55-openiscsi.rules 规则文件，将以下内容保存到文件中：

#/etc/udev/rules.d/55-openiscsi.rules

KERNEL=="sd*", BUS=="scsi", PROGRAM="/etc/udev/scripts/iscsidev.sh %b",SYMLINK+="iscsi/%c/part%n"

以上规则的含义是：为设备名以“sd”开头、设备的总线类型为 scsi 的设备创建链接文件。

PROGRAM 参数包含的是一条命令，SYMLINK 中的%c 代表的是 PROGRAM 命令的输出结果，

SYMLINK 参数表示链接文件存储的位置以及文件名，链接文件存放在/dev/iscsi/ <PROGRAM 参数中命令的执行结果>目录，文件名是 part<设备内核号>，由于所有的设备文件都存放在/dev/目录下，所以在这个设置前面没有指定/dev 目录。

下面是 PROGRAM 中指定的 iscsidev.sh 脚本。

```
#!/bin/sh
# FILE: /etc/udev/scripts/iscsidev.sh
BUS=${1}
HOST=${BUS%%:*}
[ -e /sys/class/iscsi_host ] || exit 1
file="/sys/class/iscsi_host/host${HOST}/device/session*/iscsi_session*/targetname"
target_name=$(cat ${file})
# This is not an open-scsi drive
if [ -z "${target_name}" ]; then
   exit 1
fi
echo "${target_name##*.}"
```

以上脚本的含义是将表 2-6 中 iSCSI Target IQN 的最后一个小数点之后的名称提取出来，作为 udev 创建链接文件的目录。由于 iSCSI Target 的名称是唯一的，通过这种方式就能使所有节点通过 udev 新创建的链接文件指向相同的 iSCSI 磁盘。

将刚才创建的 shell 脚本修改为可执行文件：

```
# chmod 755 /etc/udev/scripts/iscsidev.sh
```

相应的配置工作结束之后，需要重启服务器链接文件才能生成。其实，配置 udev 和在生产环境中安装多路径软件的效果是相同的，都是生成相应设备的链接文件，通过对链接文件的访问实现对设备的访问。重启服务器之后生成如下的文件结构。

```
[root@rhel1 iscsi]# pwd    /dev/iscsi

 [root@rhel1 iscsi]# tree

.   |-- dbfile1   |    `-- part -> ../../sda

    |-- dbfile2   |    `-- part -> ../../sdd

    |-- fra1   |    `-- part -> ../../sdb   |-- ocrvdisk1   |    `-- part -> ../../sdf   |-- ocrvdisk2   |    `-- part
-> ../../sdc   `-- ocrvdisk3         `-- part -> ../../sde   6 directories, 6 files
```

节点 2 执行结果：

```
[root@rhel2 iscsi]# pwd  /dev/iscsi

[root@rhel2 iscsi]# tree  .  |-- dbfile1

|   `-- part -> ../../sdf  |-- dbfile2  |   `-- part -> ../../sdc  |-- fra1  |   `-- part -> ../../sda  |-- ocrvdisk1
|   `-- part -> ../../sde  |-- ocrvdisk2  |   `-- part -> ../../sdb  `-- ocrvdisk3       `-- part -> ../../sdd
```

上面的执行结果 dbfile1、dbfile2、fra1、ocrvdisk1、ocrvdisk2、ocrvdisk3 表示的是目录，目录下面的 part 才是链接文件，链接文件对应的 sd*就是链接对象。两个节点相同路径的链接文件对应的设备文件名不是完全相同的，但它们对应的共享磁盘确是完全一致的，不同节点对相同链接文件的访问能够确保访问的是相同的共享存储。

链接文件是新生成的，并没有改动原有的以“sd”开头的设备文件，所以不同节点的/dev 目录下以“sd”开头的设备文件依然不能正确地对应 iSCSI 磁盘。