by Heinz Mauelshagen and Matthew O'Keefe

Storage technology plays a critical role in increasing the performance, availability, and manageability of Linux servers. One of the most important new developments in the Linux 2.6 kernel—on which the Red Hat® Enterprise Linux® 4 kernel is based—is the Linux Logical Volume Manager, version 2 (or LVM 2). It combines a more consistent and robust internal design with important new features including volume mirroring and clustering, yet it is upwardly compatible with the original Logical Volume Manager 1 (LVM 1) commands and metadata. This article summarizes the basic principles behind the LVM and provide examples of basic operations to be performed with it.

Introduction

Logical volume management is a widely-used technique for deploying logical rather than physical storage. With LVM, "logical" partitions can span across physical hard drives and can be resized (unlike traditional ext3 "raw" partitions). A physical disk is divided into one or more physical volumes (Pvs), and logical volume groups (VGs) are created by combining PVs as shown in Figure 1. LVM internal organization (#fig-internal). Notice the VGs can be an aggregate of PVs from multiple physical disks.

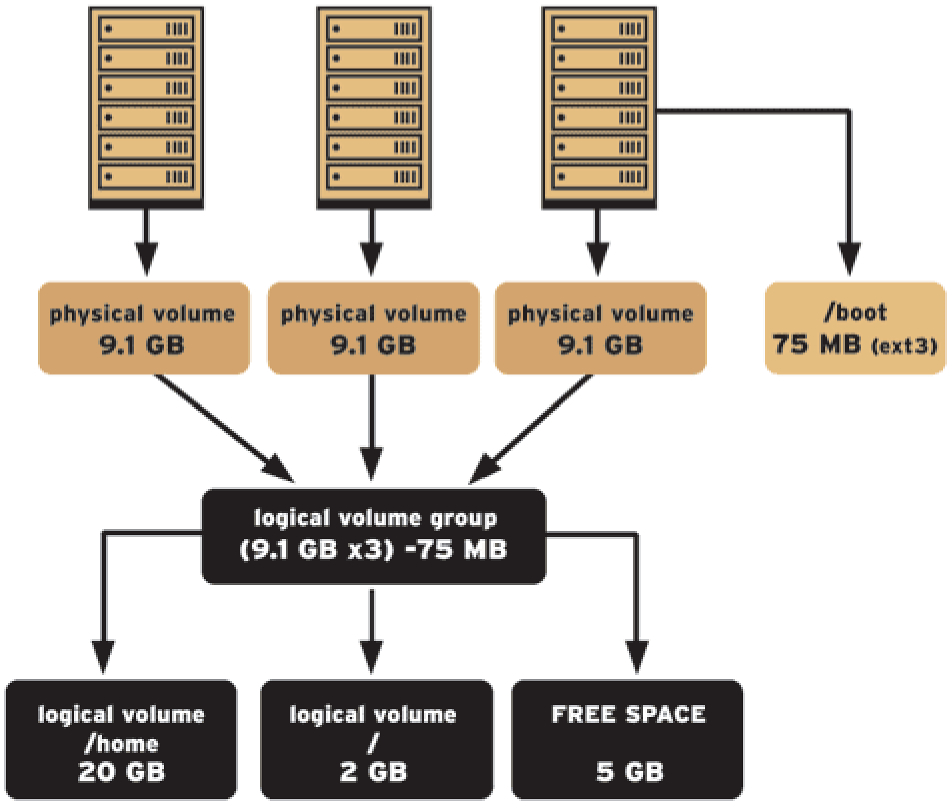

Figure 1. LVM internal organization

Figure 2. Mapping logical extents to physical extents (#fig-mapping) shows how the logical volumes are mapped onto physical volumes. Each PV consists of a number of fixed-size physical extents (PEs); similarly, each LV consists of a number of fixed-size logical extents (LEs). (LEs and PEs are always the same size, the default in LVM 2 is 4 MB.) An LV is created by mapping logical extents to physical

extents, so that references to logical block numbers are resolved to physical block numbers. These mappings can be constructed to achieve particular performance, scalability, or availability goals.

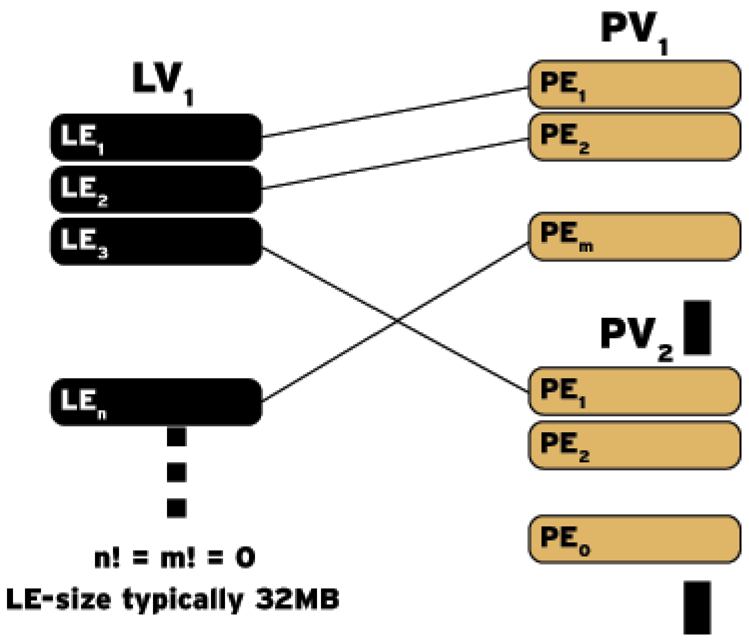

Figure 2. Mapping logical extents to physical extents

For example, multiple PVs can be connected together to create a single large logical volume as shown in Figure 3. LVM linear mapping (#fig-linear). This approach, known as a linear mapping, allows a file system or database larger than a single volume to be created using two physical disks. An alternative approach is a striped mapping, in which stripes (groups of contiguous physical extents) from alternate PVs are mapped to a single LV, as shown in Figure 4. LVM striped mapping (#fig-striped). The striped mapping allows a single logical volume to nearly achieve the combined performance of two PVs and is used quite often to achieve high-bandwidth disk transfers.

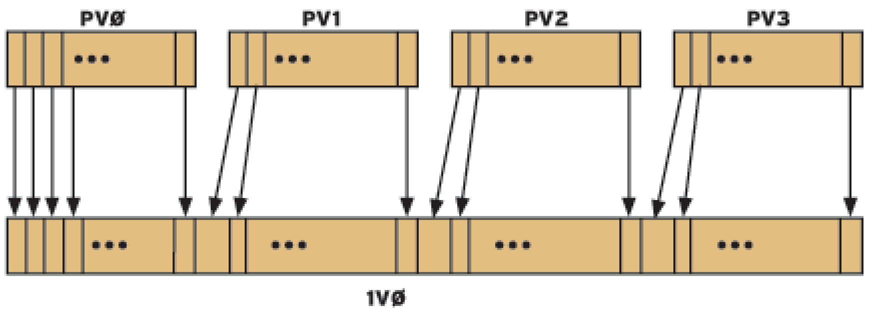

Figure 3. LVM linear mapping

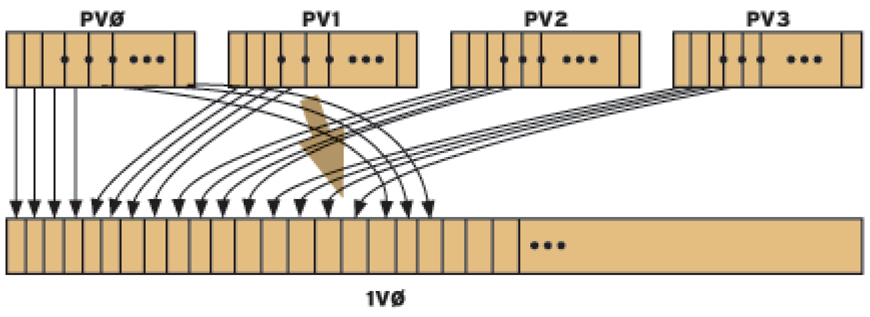

Figure 4. LVM striped mapping (4 physical extents per stripe)

Through these different types of logical-to-physical mappings, LVM can achieve four important advantages over raw physical partitions:

1. Logical volumes can be resized while they are mounted and accessible by the database or file system, removing the downtime associated with adding or deleting storage from a Linux server
2. Data from one (potentially faulty or damaged) physical device may be relocated to another device that is newer, faster or more resilient, while the original volume remains online and accessible
3. Logical volumes can be constructed by aggregating physical devices to increase performance (via disk striping) or redundancy (via disk mirroring and I/O multipathing)
4. Logical volume snapshots can be created to represent the exact state of the volume at a certain point-in-time, allowing accurate backups to proceed simultaneously with regular system operation

Basic LVM commands

Initializing disks or disk partitions
To use LVM, partitions and whole disks must first be converted into physical volumes (PVs) using the **`pvcreate`** command. For example, to convert **`/dev/hda`** and **`/dev/hdb`** into PVs use the following commands:

```
pvcreate /dev/hda
pvcreate /dev/hdb
```

If a Linux partition is to be converted make sure that it is given partition type 0x8E using **`fdisk`**, then use **`pvcreate`**:

```
pvcreate /dev/hda1
```

Creating a volume group
Once you have one or more physical volumes created, you can create a volume group from these PVs using the **`vgcreate`** command. The following command:

```
vgcreate  volume_group_one /dev/hda /dev/hdb
```

creates a new VG called **`volume_group_one`** with two disks, **`/dev/hda`** and **`/dev/hdb`**, and 4 MB PEs. If both **`/dev/hda`** and **`/dev/hdb`** are 128 GB in size, then the VG **`volume_group_one`** will have a total of 2**16 physical extents that can be allocated to logical volumes.

Additional PVs can be added to this volume group using the **`vgextend`** command. The following commands convert **`/dev/hdc`** into a PV and then adds that PV to **`volume_group_one`**:

```
pvcreate /dev/hdc
vgextend volume_group_one /dev/hdc
```

This same PV can be removed from **volume_group_one** by the **vgreduce** command:

```
vgreduce volume_group_one /dev/hdc
```

Note that any logical volumes using physical extents from PV **/dev/hdc** will be removed as well. This raises the issue of how we create an LV within a volume group in the first place.

Creating a logical volume
We use the **lvcreate** command to create a new logical volume using the free physical extents in the VG pool. Continuing our example using VG volume_group_one (with two PVs **/dev/hda** and **/dev/hdb** and a total capacity of 256 GB), we could allocate nearly all the PEs in the volume group to a single linear LV called **logical_volume_one** with the following LVM command:

```
lvcreate -n logical_volume_one   --size 255G volume_group_one
```

Instead of specifying the LV size in GB we could also specify it in terms of logical extents. First we use **vgdisplay** to determine the number of PEs in the volume_group_one:

```
vgdisplay volume_group_one | grep "Total PE"
```

which returns

```
Total PE   65536
```

Then the following **lvcreate** command will create a logical volume with 65536 logical extents and fill the volume group completely:

```
lvcreate -n logical_volume_one  -l 65536 volume_group_one
```

To create a 1500MB linear LV named **logical_volume_one** and its block device special file **/dev/volume_group_one/logical_volume_one** use the following command:

```
lvcreate -L1500 -n logical_volume_one volume_group_one
```

The **lvcreate** command uses linear mappings by default.

Striped mappings can also be created with **lvcreate**. For example, to create a 255 GB large logical volume with two stripes and stripe size of 4 KB the following command can be used:

```
lvcreate -i2 -I4 --size 255G -n logical_volume_one_striped volume_group_one
```

It is possible to allocate a logical volume from a specific physical volume in the VG by specifying the PV or PVs at the end of the **lvcreate** command. If you want the logical volume to be allocated from a specific physical volume in the volume group, specify the PV or PVs at the end of the **lvcreate** command line. For example, this command:

```
lvcreate -i2 -I4 -L128G -n logical_volume_one_striped volume_group_one /dev/hda /dev/hdb
```

creates a striped LV named **logical_volume_one** that is striped across two PVs (**/dev/hda** and **/dev/hdb**) with stripe size 4 KB and 128 GB in size.

An LV can be removed from a VG through the **lvremove** command, but first the LV must be unmounted:

```
umount /dev/volume_group_one/logical_volume_one
lvremove /dev/volume_group_one/logical_volume_one
```

Note that LVM volume groups and underlying logical volumes are included in the device special file directory tree in the **/dev** directory with the following layout:

```
/dev/<volume_group_name>/<logical_volume_name>
```

so that if we had two volume groups **myvg1** and **myvg2** and each with three logical volumes named

lv01, **lv02**, **lv03**, six device special files would be created:

```
/dev/myvg1/lv01
/dev/myvg1/lv02
/dev/myvg1/lv03
/dev/myvg2/lv01
/dev/myvg2/lv02
/dev/myvg2/lv03
```

Extending a logical volume
An LV can be extended by using the lvextend command. You can specify either an absolute size for the extended LV or how much additional storage you want to add to the LVM. For example:

```
lvextend -L120G /dev/myvg/homevol
```

will extend LV **/dev/myvg/homevol** to 12 GB, while

```
lvextend -L+10G /dev/myvg/homevol
```

will extend LV **/dev/myvg/homevol** by an additional 10 GB. Once a logical volume has been extended, the underlying file system can be expanded to exploit the additional storage now available on the LV. With Red Hat Enterprise Linux 4, it is possible to expand both the ext3fs and GFS file systems online, without bringing the system down. (The ext3 file system can be shrunk or expanded offline using the **ext2resize** command.) To resize ext3fs, the following command

```
ext2online /dev/myvg/homevol
```

will extend the ext3 file system to completely fill the LV, **/dev/myvg/homevol**, on which it resides.

The file system specified by device (partition, loop device, or logical volume) or mount point must currently be mounted, and it will be enlarged to fill the device, by default. If an optional size parameter is specified, then this size will be used instead.

Differences between LVM1 and LVM2
The new release of LVM, LVM 2, is available only on Red Hat Enterprise Linux 4 and later kernels. It is upwardly compatible with LVM 1 and retains the same command line interface structure. However it uses a new, more scalable and resilient metadata structure that allows for transactional metadata updates (that allow quick recovery after server failures), very large numbers of devices, and clustering.

For Enterprise Linux servers deployed in mission-critical environments that require high availability, LVM2 is the right choice for Linux volume management. Table 1. A comparison of LVM 1 and LVM 2 (#tb-compare) summarizes the differences between LVM1 and LVM2 in features, kernel support, and other areas.

Features	LVM1	LVM2
RHEL AS 2.1 support	No	No
RHEL 3 support	Yes	No
RHEL 4 support	No	Yes
Transactional metadata for fast recovery	No	Yes
Shared volume mounts with GFS	No	Yes
Cluster Suite failover supported	Yes	Yes
Striped volume expansion	No	Yes
Max number PVs, LVs	256 PVs, 256 LVs	2**32 PVs, 2**32 LVs
Max device size	2 Terabytes	8 Exabytes (64-bit CPUs)
Volume mirroring support	No	Yes, in Fall 2005

Table 1. A comparison of LVM 1 and LVM 2

Summary

The Linux Logical Volume Manager provides increased manageability, uptime, and performance for Red Hat Enterprise Linux servers. You can learn more about LVM by visiting to following websites:

- **LVM HOWTO** (http://www.tldp.org/HOWTO/LVM-HOWTO/)
- **LVM2 Resource Page** (http://sources.redhat.com/lvm2/)

About the authors

From 1990 to May 2000, Matthew O'Keefe taught and performed research in storage systems and parallel simulation software as a professor of electrical and computer engineering at the University of Minnesota. He founded Sistina Software in May of 2000 to develop storage infrastructure software for Linux, including the Global File System (GFS) and the Linux Logical Volume Manager (LVM). Sistina was acquired by Red Hat in December 2003, where Matthew now directs storage software strategy.

Starting in 2000, Heinz Mauelshagen began working on device mapper and LVM at Sistina Software. Sistina was acquired by Red Hat in December 2003. Heinz Mauelshagen is currently continuing his work on clustering and storage as a Red Hat developer in Germany. Before joining Sistina, Heinz was a senior system administrator at T-Systems for a decade.

- Connect:
-
-