

EE411: Fault Diagnostic and Tolerant Systems
Instructor: DR M S SHAIKH

LOGICAL VOLUME MANAGEMENT

Department of Telecommunication and Computer Engineering
National University of Computer and Emerging Sciences
Karachi Campus

December 23, 2006

Abstract

Abstraction is an important programming concept. By hiding (abstracting) the inner workings and details of modules and data structures, the use of such similar entities is made simple. The programmer, making use of these entities, is relieved from writing code that is tightly bound to the internal representation of the entities. As long as a module carries a consistent interface, changes in its internal details do not break interoperability between code that uses it. An extremely complex piece of routine looks simple to users. Such is the power of abstraction. But that sort of abstraction applies to code.

Imagine abstraction at the storage level. Imagine separating the hardware details of storage devices from the software management in such a manner that changes in hardware do not affect applications using the storage. Imagine adding disks, resizing space at run-time, without having to halt any application, stop the system. How powerful such an abstraction can be, if it exists.

LOGICAL VOLUME MANAGEMENT: The powerful abstraction of mass storage devices to present a hardware-independent view of storage to applications.

This report introduces LOGICAL VOLUME MANAGEMENT from the point of view of storage abstraction, delves into the details of how LOGICAL VOLUME MANAGEMENT works, concluding with a comparison and contrast of LOGICAL VOLUME MANAGEMENT with standard RAID technology.

0.1 Introduction

LOGICAL VOLUME[1] is an abstraction of a physical hard-disk, a (set of) partition(s) in a physical hard-disk, a set of physical hard-disks concatenated together in a standard RAID setting, or any combination of these. LOGICAL VOLUME MANAGEMENT (LVM)[2] is management of any number of LOGICAL VOLUMES over mass storage to provide to applications and the Operating System a hardware-independent view of the storage subsystem.

Delicately defined in the Whitepaper on LVM[3], Volume Management creates a layer of abstraction over the physical storage system so that applications access a virtual storage which is independent of the Hardware on which the bits are actually stored. Not only are the details of where individual bits are stored and on which partitions of which hard-disk(s) separated, LOGICAL VOLUME MANAGEMENT brings a host of advantages not addressable through RAID or standard non-LVM storage systems, including, by far the most promising benefit, resizing of storage at *run-time*, without the need to shut down applications or the system.

0.2 How LVM works

In order to understand how LOGICAL VOLUME MANAGEMENT works, the following terminologies need be clarified first:

Physical Volume (PV) A hard-disk, a partition or set of partitions within a hard-disk, a set of hard-disks, a Raid setting.

Volume Group (VG) A container for any number of physical volumes to represent the collection as one big logical hard-disk.

Logical Volume (LV) A logical equivalent of partitions on hard-disks, contained within a particular volume group, where the actual data are stored. A single volume group contains a number of logical volumes.

Physical Extent (PE) The smallest physical storage unit in a Logical Volume Management system. PEs are unique within VGs, and are where the actual bits are stored within LVs. They are measured as being within VGs in quantities determined by the mathematical expression: $VG.size / PE.size = PE.count$

Logical Extent (LE) Also the smallest physical storage unit and about the same size as that of PE. Unlike PEs, LEs are unique within LVs, so they are logical equivalents of PEs. Like PEs, they are measured as being within LVs in quantities determined by the mathematical expression: $LV.size / LE.size^1 = LE.count$.

¹Sizes of individual LEs and PEs are the same.

Physical Volumes are at the heart of any storage system. PVs represent the actual physical storage devices, which can be a (set of) hard-disk(s), a (set of) partition(s), or several RAID configurations. In an LVM system, the Physical Volumes are grouped together into Volume Groups which portray the individual Physical Volumes as a single, or multiple, big Logical hard-disk(s). Logical Volumes are logical equivalents of partitions on a hard-disk. Volume Groups are collections of Logical Volumes.

In detail, each Physical Volume is divided into Physical Extents². Each Physical Extent enjoys a unique ID within the Volume Group it exists in. Each Logical Volume, in turn, is chopped down into equally-sized Logical Extents³. While PEs are unique globally within a Volume Group, LEs, equivalently, are unique globally within a Logical Volume.

This metadata information, which includes the PEs, LEs, VGs, PVs, and LVs for each physical device, is stored on the starting sector of each physical device, and is read by the LVM subsystem on boot-up to setup LVM resources.

When access to a logical storage is desired, the LE of the space is identified, and the ID of LE is indexed into a mapping table for the Physical Volume in which the Logical Volume in which the identified Logical Extent exists rests to find the PE and PV information.

Bryce Harrington and Kees Cook, in their article *Managing Disks with LVM*[4], guide through a step-by-step tutorial on setting up LVM on Linux. Also, the *LVM HOWTO*[5] describes in gory details the process of setting up LVM on Linux.

0.3 What LVM can do

LOGICAL VOLUME MANAGEMENT brings about a number of benefits not conceivable with standard storage technologies, including RAID. A few of the features of LOGICAL VOLUME MANAGEMENT are introduced in this section.

LVM separates the hardware details of the storage system from software management. Applications look at and talk to logical volumes or logical storage. They are not concerned, do not know, and should not bother to know, where bits are stored, on which partition of which hard-disk. As far as the LVM system is concerned, bits are stored in corresponding LEs and PEs. A collection of LEs forms an LV, collections of which form a VG. Applications read from and write to LVs in VGs. A side-effect of this distributed setup is that the size of any VG can be increased by increasing the number of LVs in it, and the size of any LV can be increased by increasing the LEs it contains. This brings in the ability to resize Logical Volumes and Volume Groups. However, LVM makes it more flexible than this. *With LVM, resizing of LVs and VGs happens on the fly,*

²The size of PE is variable, but within one Volume Group, all the PEs are of a fixed size (usually 4-MB).

³LEs within the LV are of the same size as that of PEs which are within the VG in which that LV is.

at run-time, all the while applications are reading from and writing to logical storage.

Snapshots is another powerful feature LVM brings with itself. At any time, a snapshot of a VG can be taken. The snapshot is a frozen, read-only copy of the VG, which can be utilised for backup and recovery purposes. Applications continue to read from and write to the original VG, while the snapshot stays frozen, open to read-only access.

LOGICAL VOLUME MANAGEMENT also supports RAID-like concatenation and striping of disks. In normal storage systems and RAID, the location of physical hard-disks matters a lot. With LVM, a storage capacity or device can be anywhere, and can exist independent of any other device. LVM stores the meta information on the device, and reads it on boot time, and reconstructs the LVM resources on the fly, disregarding the physical arrangement of hard-disks and the order in which they are detected on boot-up.

0.4 LVM and RAID

Few distinguishing features of LOGICAL VOLUME MANAGEMENT have already been described in the previous section (0.3). It should be noted that LVM is no replacement of RAID. And like RAID, it is no replacement of regular backups.

LOGICAL VOLUME MANAGEMENT works independent of any storage system. Hardware RAID, in contrast, is dependent on an electronic controller, while Software RAID, in the absence of hardware, does a software emulation of the hardware controller. LVM is completely independent of hardware. It can work with bare, standard storage devices, or any complex RAID configurations.

The high-availability provided by LVM allows for run-time resizing, moving, and duplication of storage systems without the need of shutting down the systems. In fact, the applications are not required to halt, and can continue reading from and writing to logical store.

RAID comes with expensive complex software, but the software is limited in its ability only to handle the RAID array and simulate a hardware RAID controller. LVM, on the other hand, is *completely free* and *completely independent* of the underlying storage system is abstracts.

LVM can also provide RAID-like striping, and mirroring functions, plus concatenation of disks (such a JBOD).

0.5 Conclusion

LOGICAL VOLUME MANAGEMENT is an excellent means of managing storage. It brings in a lot of benefits, some surpassing those provided by present RAID technology. It is easy to setup. And most of all, as with LINUX, it comes *absolutely free of cost*.

However, LOGICAL VOLUME MANAGEMENT does not provide protection against disk failures. Like RAID, it *should not* be considered a substitute for

a complete backup recovery system. Nothing can beat backups in cases of disk failures.

Bibliography

- [1] Wikipedial Logical Volume http://en.wikipedia.org/wiki/Logical_volume
- [2] Wikipedia: Logical Volume Management http://en.wikipedia.org/wiki/Logical_volume_management
- [3] Whitepaper: Logical Volume Management (LVM) http://www.idevelopment.info/data/Unix/Linux/LINUX_lvm_whitepaper_SuSE.pdf
- [4] Linuxdevcenter.com: Managing Disk Space with LVM <http://www.linuxdevcenter.com/pub/a/linux/2006/04/27/managing-disk-space-with-lvm.html>
- [5] TLDP.org: LVM HOWTO <http://tldp.org/HOWTO/LVM-HOWTO/index.html>