Logical Volume Manager HOWTO

Table of Contents

Table of Contents

Logical Volume Manager HOWTO

bert hubert <ahu@ds9a.nl>
Richard Allen <ra@ra.is>

Version 0.0.2 $Date: 2000/04/28 01:27:32 $

A very hands−on HOWTO for Linux LVM

5.Growing your filesystem

6.Replacing disks

7.Making snapshots for consistent backups

8.Redundancy & Performance

9.Cookbook

10.Further reading

11.Acknowledgements & Thanks to

1.Introduction

Welcome, gentle reader.

This document is written to help enlighten you on what LVM is, how it works, and how you can use it to make your life easier. While there is an LVM FAQ, and even a German HOWTO, this document is written from a different perspective. It is a true 'HOWTO' in that it is very hands−on, while also imparting understanding (hopefully).

I should make it clear that I am not an author of the Linux Logical Volume Manager. I have great respect for the people who are, and hope to be able to cooperate with them.

It's even weirder, I don't even know the developers of LVM. I hope this will change soon. I apologise in advance for stepping on peoples toes.

1.1 Disclaimer & License

This document is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

If your disks melt and your company fires you − it's never our fault. Sorry. Make frequent backups and do your experiments on non−mission critical systems.

Furthermore, Richard Allen does not speak for his employer.

Linux is a registered trademark of Linus Torvalds.

1.2 Prior knowledge

Not much. If you have ever installed Linux and made a filesystem (fdisk/mkfs), you should be all set. As always when operating as root, caution is advised. Incorrect commands or any operation on device files may damage your existing data.

If you know how to configure HP/UX LVM you are almost done, Linux works almost exactly like the HP implementation.

1.3 Housekeeping notes

There are several things which should be noted about this document. While I wrote most of it, I really don't want it to stay that way. I am a strong believer in Open Source, so I encourage you to send feedback, updates, patches etcetera. Do not hesitate to inform us of typos or plain old errors.

If you feel to you are better qualified to maintain a section, or think that you can author and maintain new sections, you are welcome to do so. The SGML of this HOWTO is available via CVS. I envision this being a collaborative project.

In aid of this, you will find lots of FIXME notices. Patches are always welcome! Wherever you find a

FIXME, you should know that you are treading unknown territory. This is not to say that there are no errors elsewhere, but be extra careful. If you have validated something, please let us know so I can remove the FIXME notice.

1.4 Access, CVS & submitting updates

The canonical location for the HOWTO is http://www.ds9a.nl/lvm–howto/.

We now have anonymous CVS access available for the world at large. This allows you to easily obtain the latest version of this HOWTO and to provide your changes and enhancements.

If you want to grab a copy of the HOWTO via CVS, here is how to do so:

```
$ export CVSROOT=:pserver:anon@outpost.ds9a.nl:/var/cvsroot
$ cvs login
CVS password: [enter 'cvs' (without 's)]
$ cvs co lvm-howto
cvs server: Updating lvm-howto
U lvm-howto/lvm-howto.sgml
```

If you spot an error, or want to add something, just fix it locally, and run "cvs diff –u", and send the result off to us.

A Makefile is supplied which should help you create postscript, dvi, pdf, html and plain text. You may need to install sgml–tools, ghostscript and tetex to get all formats.

1.5 Layout of this document

We will initially be explaining some basic stuff which is needed to do things. We do try however to include examples where this would aid comprehension.

2.What is LVM?

Historically, a partition size is static. This requires a system installer to have to consider not the question of "how much data will I store on this partition", but rather "how much data will I *EVER* store on this partition". When a user runs out of space on a partition, they either have to re–partition (which may involve an entire operating system reload) or use kludges such as symbolic links.

The notion that a partition was a sequential series of blocks on a physical disc has since evolved. Most Unix–like systems now have the ability to break up physical discs into some number of units. Storage units from multiple drives can be pooled into a "logical volume", where they can be allocated to partitions. Additionally, units can be added or removed from partitions as space requirements change.

This is the basis of a Logical Volume Manager (LVM).

For example, say that you have a 1GB disc and you create the "/home" partition using 600MB. Imagine that you run out of space and decide that you need 1GB in "/home". Using the old notion of partitions, you'd have to have another drive at least 1GB in size. You could then add the disc, create a new /home, and copy the existing data over.

However, with an LVM setup, you could simply add a 400MB (or larger) disc, and add it's storage units to the "/home" partition. Other tools allow you to resize an existing file−system, so you simply resize it to take advantage of the larger partition size and you're back in business.

As a very special treat, LVM can even make 'snapshots' of itself which enable you to make backups of a non−moving target. We return to this exciting possibility, which has lots of other real−world applications, later on.

In the next section we explain the basics of LVM, and the multitude of abstractions it uses.

3.Basic principles

Ok, don't let this scare you off, but LVM comes with a lot of jargon which you should understand lest you endanger your filesystems.

We start at the bottom, more or less.

The physical media

> You should take the word 'physical' with a grain of salt, though we will initially assume it to be a simple hard disk, or a partition. Examples, /dev/hda, /dev/hda6, /dev/sda. You can turn any consecutive number of blocks on a block device into a ...

Physical Volume (PV)

> A PV is nothing more than a physical medium with some administrative data added to it − once you have added this, LVM will recognise it as a holder of ...

Physical Extents (PE)

> Physical Extents are like really big blocks, often with a size of megabytes. PEs can be assigned to a...

Volume Group

> A VG is made up of a number of Physical Extents (which may have come from multiple Physical Volumes or hard drives). While it may be tempting to think of a VG as being made up of several hard drives (/dev/hda and /dev/sda for example), it's more accurate to say that it contains PEs which are provided by these hard drives.

>From this Volume Group, PEs can be assigned to a ...

Logical Volume (LV)

Yes, we're finally getting somewhere. A Logical Volume is the end result of our work, and it's there that we store our information. This is equivalent to the historic idea of partitions. As with a regular partition, on this Logical Volume you would typically build a ...

Filesystem

This filesystem is whatever you want it to be: the standard ext2, ReiserFS, NWFS, XFS, JFX, NTFS, etc... To the linux kernel, there is no difference between a regular partition and a Logical Volume.

I've attempted some ASCII art which may help you visualise this.

```
A Physical Volume, containing Physical Extents:

  +-----[ Physical Volume ]------+
  | PE | PE | PE | PE | PE | PE  |
  +------------------------------+

A Volume Group, containing 2 Physical Volumes (PVs) with 6 Physical Extents:

  +------[ Volume Group ]-----------------+
  |  +--[PV]--------+  +--[PV]---------+  |
  |  | PE | PE | PE |  | PE | PE | PE  |  |
  |  +--------------+  +---------------+  |
  +---------------------------------------+

We now further expand this:

  +------[ Volume Group ]-----------------+
  |  +--[PV]--------+  +--[PV]---------+  |
  |  | PE | PE | PE |  | PE | PE | PE  |  |
  |  +--+---+---+---+  +-+----+----+---+  |
  |     |   |   | +-----/     |    |      |
  |     |   |   | |           |    |      |
  |   +-+---+---+-+      +----+----+--+   |
  |   |  Logical  |      |  Logical   |   |
  |   |  Volume   |      |   Volume   |   |
  |   |           |      |            |   |
  |   |  /home    |      |    /var    |   |
  |   +-----------+      +------------+   |
  +---------------------------------------+
```

This shows us two filesystems, spanning two disks. The /home filesystem contains 4 Physical Extents, the /var filesystem 2.

bert hubert is writing a tool to represent LVM more visually, a screenshot is provided. Looks better than the ASCII art.

3.1 Show & Tell

Ok, this stuff is hard to assimilate ('We are LVM of Borg...'), so here is a very annotated example of creating a Logical Volume. Do NOT paste this example onto your console because you WILL destroy data, unless it happens that on your computer /dev/hda3 and /dev/hdb2 aren't used.

When in doubt, view the ASCIIgram above.

You should first set the partition types of /dev/hda3 and /dev/hdb2 to 0x8e, which is 'Linux LVM'. Please note that your version of fdisk may not yet know this type, so it will be listed as 'Unknown':

```
# fdisk /dev/hda

Command (m for help): p

Disk /dev/hda: 255 heads, 63 sectors, 623 cylinders
Units = cylinders of 16065 * 512 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/hda1             1         2     16033+  83  Linux
/dev/hda2             3       600   4803435   83  Linux
/dev/hda3           601       607     56227+  83  Linux
/dev/hda4           608       614     56227+  83  Linux

Command (m for help): t
Partition number (1-4): 3
Hex code (type L to list codes): 8e

Command (m for help): p

Disk /dev/hda: 255 heads, 63 sectors, 623 cylinders
Units = cylinders of 16065 * 512 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/hda1             1         2     16033+  83  Linux
/dev/hda2             3       600   4803435   83  Linux
/dev/hda3           601       607     56227+  8e  Unknown
/dev/hda4           608       614     56227+  83  Linux

Command (m for help): w
```

We do the same for /dev/hdb2, but we don't display it here. This is needed so that LVM is able to reconstruct things should you lose your configuration.

Now, this shouldn't be necessary, but some computers require a reboot at this point. So if the following examples don't work, try rebooting.

We then create our Physical Volumes, like this:

```
# pvcreate  /dev/hda3
pvcreate -- physical volume "/dev/hda3" successfully created
```

```
# pvcreate  /dev/hdb2
pvcreate -- physical volume "/dev/hdb2" successfully created
```

We than add these two PVs to a Volume Group called 'test':

```
# vgcreate test /dev/hdb2 /dev/hda3
vgcreate -- INFO: using default physical extent size 4 MB
vgcreate -- INFO: maximum logical volume size is 255.99 Gigabyte
vgcreate -- doing automatic backup of volume group "test"
vgcreate -- volume group "test" successfully created and activated
```

So we now have an empty Volume Group, let's examine it a bit:

```
# vgdisplay -v test
--- Volume group ---
VG Name               test
VG Access             read/write
VG Status             available/resizable
VG #                  0
MAX LV                256
Cur LV                0
Open LV               0
MAX LV Size           255.99 GB
Max PV                256
Cur PV                2
Act PV                2
VG Size               184 MB
PE Size               4 MB
Total PE              46
Alloc PE / Size       0 / 0
Free  PE / Size       46 / 184 MB

--- No logical volumes defined in test ---


--- Physical volumes ---
PV Name (#)           /dev/hda3 (2)
PV Status             available / allocatable
Total PE / Free PE    13 / 13

PV Name (#)           /dev/hdb2 (1)
PV Status             available / allocatable
Total PE / Free PE    33 / 33
```

Lots of data here – most of it should be understandable by now. We see that there are no Logical Volumes defined, so we should work to remedy that. We try to generate a 50 megabyte volume called 'HOWTO' in the Volume Group 'test':

```
# lvcreate -L 50M -n HOWTO test
lvcreate -- rounding up size to physical extent boundary "52 MB"
lvcreate -- doing automatic backup of "test"
lvcreate -- logical volume "/dev/test/HOWTO" successfully created
```

Ok, we're nearly there, let's make a filesystem:

```
# mke2fs /dev/test/HOWTO
mke2fs 1.18, 11-Nov-1999 for EXT2 FS 0.5b, 95/08/09
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
13328 inodes, 53248 blocks
2662 blocks (5.00%) reserved for the super user
First data block=1
7 block groups
8192 blocks per group, 8192 fragments per group
1904 inodes per group
Superblock backups stored on blocks:
        8193, 24577, 40961

Writing inode tables: done
Writing superblocks and filesystem accounting information: done
# mount /dev/test/HOWTO /mnt
# ls /mnt
lost+found
```

And we're done! Let's review our Volume Group, because it should be filled up a bit by now:

```
# vgdisplay test -v
--- Volume group ---
VG Name               test
VG Access             read/write
VG Status             available/resizable
VG #                  0
MAX LV                256
Cur LV                1
Open LV               1
MAX LV Size           255.99 GB
Max PV                256
Cur PV                2
Act PV                2
VG Size               184 MB
PE Size               4 MB
Total PE              46
Alloc PE / Size       13 / 52 MB
Free  PE / Size       33 / 132 MB

--- Logical volume ---
LV Name               /dev/test/HOWTO
VG Name               test
LV Write Access       read/write
LV Status             available
LV #                  1
# open                1
LV Size               52 MB
Current LE            13
Allocated LE          13
Allocation            next free
```

```
Read ahead sectors    120
Block device          58:0


--- Physical volumes ---
PV Name (#)           /dev/hda3 (2)
PV Status             available / allocatable
Total PE / Free PE    13 / 13

PV Name (#)           /dev/hdb2 (1)
PV Status             available / allocatable
Total PE / Free PE    33 / 20
```

Well, it is. /dev/hda3 is completely unused, but /dev/hdb2 has 13 Physical Extents in use.

3.2 Active and Inactive: kernel space and user space

As with all decent operating systems, Linux is divided in two parts: kernel space and user space. Userspace is sometimes called userland, which would also be a good name for a theme park, 'Userland'.

Discovery, creation and modification of things pertaining to Logical Volume Management is done in user space, and then communicated to the kernel. Once a volume group or logical volume is reported to the kernel, it is called 'Active'. Certain changes can only be performed when an entity is active, others only when it is not.

4.Prerequisites

There is a wide range of kernels where LVM is available on. In Linux 2.4, LVM will be fully integrated. From kernel 2.3.47 and onwards, LVM is in the process of being merged into the main kernel.

4.1 Kernel

Linux 2.4

Will contain everything you need. It is expected that most distributions will release with LVM included as a module. If you need to compile, just tick off the LVM option when selecting your block devices.

Linux 2.3.99.*

Once things have calmed down on the kernel development front, this section will vanish. For now, the gory details.

As we write this, Linux 2.3.99pre5 is current and it still needs a very tiny patch to get LVM working.

For Linux 2.3.99pre3, two patches were released:

The patch was posted on linux−kernel, and is available here.

Andrea Arcangeli improved on that patch, and supplied an incremental patch, which should be applied on top of the 2.3.99pre3 LVM patch above.

For Linux 2.3.99pre5, bert hubert rolled these two patches into one and ported it to 2.3.99pre5. Patch. Use with care.

2.3.99pre6−1, yes, a prerelease of a prepatch, features for the first time complete LVM support! It stil misses Andreas patch but we have been assured that it is in the queue to be released real soon.

2.3.99pre4−ac1 has the tiny LVM patch in by default, and working. It does not contain Andreas patch though.

Linux 2.2

FIXME: write this

Linux 2.3

FIXME: write this

4.2 Userspace

You need the tools available from the LVM site. Compiling them on glibc2.1 systems requires a tiny patch, and even then gives errors on Debian 2.2.

5.Growing your filesystem

You can do this with a provided script which does a lot of work for you, or you can do it by hand if needed.

5.1 With e2fsadm

If there is room within your volume group, and you use the ext2 filesystem (most people do), you can use this handy tool.

The `e2fsadm` command uses the commercial resize2fs tool. While people feel that this is good software, it is not very widely installed.

If you want to use the FSF's `ext2resize` command, you need to inform `e2fsadm` of this:

```
# export E2FSADM_RESIZE_CMD=ext2resize
# export E2FSADM_RESIZE_OPTS=""
```

The rest is easy, `e2fsadm` is a lot like the other LVM commands:

```
# e2fsadm /dev/test/HOWTO -L+50M
e2fsadm -- correcting size 102 MB to physical extent boundary 104 MB
e2fsck 1.18, 11-Nov-1999 for EXT2 FS 0.5b, 95/08/09
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
/dev/test/HOWTO: 11/25688 files (0.0% non-contiguous), 3263/102400 blocks
lvextend -- extending logical volume "/dev/test/howto" to 104 MB
lvextend -- doing automatic backup of volume group "test"
lvextend -- logical volume "/dev/test/HOWTO" successfully extended

ext2_resize_fs
ext2_grow_fs
ext2_block_relocate
ext2_block_relocate_grow
ext2_grow_group
ext2_add_group
ext2_add_group
ext2_add_group
ext2_add_group
ext2_add_group
ext2_add_group
direct hits 4096 indirect hits 0 misses 1
e2fsadm -- ext2fs in logical volume "/dev/test/HOWTO" successfully extended to 104 MB
```

5.2 Growing your Logical Volume

The `e2fsadm` command takes care of this for you. However, it may be useful to understand how to do this manually:

If you have room within your Volume Group, this is a one liner:

```
# lvextend -L+12M /dev/test/HOWTO
lvextend -- rounding size to physical extent boundary
lvextend -- extending logical volume "/dev/test/HOWTO" to 116 MB
lvextend -- doing automatic backup of volume group "test"
lvextend -- logical volume "/dev/test/HOWTO" successfully extended
```

5.3 Growing your Volume Group

This is done with the vgextend utility, and is easy as pie. You first need to create a physical volume. This is done with the `pvcreate` utility. With this tool, you convert any block device into a physical volume.

After that is done, `vgextend` does the rest:

```
# pvcreate /dev/sda1
pvcreate -- physical volume "/dev/sda1" successfully created
# vgextend webgroup /dev/sda1
vgextend -- INFO: maximum logical volume size is 255.99 Gigabyte
vgextend -- doing automatic backup of volume group "webgroup"
vgextend -- volume group "webgroup" successfully extended
```

Please note that in order to do this, your Volume Group needs to be active. You can make it by executing 'vgchange −a y webgroup'.

5.4 Growing your filesystem

If you want to do this manually, there are a couple of ways to do this.

ext2 off−line with ext2resize

By off−line, we mean that you have to unmount the file−system to make these changes. The file−system and it's data will be unavailable while doing this. Note this means you must use other boot media if extending the size of the root or other important partitions.

The ext2resize tool is available on the GNU ftp size, but most distributions carry it as a package. The syntax is very straightforward:

```
# ext2resize /dev/HOWTO/small 40000
```

Where 40000 is the number of blocks the filesystem should have after growing or shrinking.

ext2 on−line

FIXME: write this

6.Replacing disks

This is one of the benefits of LVM. Once you start seeing errors on a disk, it is high time to move your data. With LVM this is easy as pie. We first do the obvious replacement example where you add a disk to the system that's at least as large as the one you want to replace.

To move data, we move Physical Extents of a Volume Group to another disk, or more precisely, to another Physical Volume. For this, LVM offers us the `pvmove` utility.

Let's say that our suspicious disk is called /dev/hda1 and we want to replace it by /dev/sdb3. We first add /dev/sdb3 to the Volume Group that contains /dev/hda1.

It appears advisable to unmount any filesystems on this Volume Group before doing this. Having a full backup might not hurt either.

FIXME: is this necessary?

We then execute `pvmove`. In its simplest invocation, we just mention the disk we want to remove, like this:

```
# pvmove /dev/hda1
pvmove -- moving physical extents in active volume group "test1"
pvmove -- WARNING: moving of active logical volumes may cause data loss!
pvmove -- do you want to continue? [y/n] y
pvmove -- doing automatic backup of volume group "test1"
pvmove -- 12 extents of physical volume "/dev/hda1" successfully moved
```

Please heed this warning. Also, it appears that at least some kernels or LVM versions have trouble with this command. I tested it with 2.3.99pre6−2, and it works, but be warned.

Now that /dev/hda1 contains no Physical Extents anymore, we can reduce it from the Volume Group:

```
# vgreduce test1 /dev/hda1
vgreduce -- doing automatic backup of volume group "test1"
vgreduce -- volume group "test1" successfully reduced by physical volume:
vgreduce -- /dev/hda1
```

FIXME: we need clarity on a few things. Should the volume group be active? When do we get data loss?

6.1 When it's too late

If a disk fails without warning and you are unable to move the Physical Extents off it to a different Physical Volume you will have lost data unless the Logical Volumes on the PV that failed was mirrored. The correct course of action is to replace the failed PV with an identical one or at least a partition of the same size.

The directory /etc/lvmconf contains backups of the LVM data and structures that make the disks into Physical Volumes and list which Volume Groups that PV belongs to and what Logical Volumes are in the Volume Group.

After replacing the faulty disk you can use the `vgcfgrestore` command to recover the LVM data to the new PV. This restores the Volume Group and all it's info, but it does not restore the data that was in the Logical Volumes. This is why most LVM commands make backups automatically of the LVM data when doing changes.

7.Making snapshots for consistent backups

This is one of the more incredible possibilities. Let's say you have a busy server, with lots of things happening. For a useful backup, you need to shut down a large number of programs because otherwise you end up with inconsistencies.

The canonical example is moving a file from /tmp to /root, where /root was being backed up first. When /root was read, the file wasn't there yet. By the time /tmp was backed up, the file was gone.

Another story goes for saving databases or directories. We have no clue if a file is in any usable state unless we give the application time to do a clean shutdown.

Which is where another problem pops up. We shut down out applications, make our backup, and restart them again. This is all fine as long as the backup only takes a few minutes, but gets to be real painful if it takes hours, or if you're not even sure how long it takes.

LVM to the rescue.

With LVM we can make a snapshot picture of a Logical Volume which is instantaneous, and then mount that and make a backup of it.

Let's try this out:

```
# mount /dev/test/HOWTO /mnt
# echo > /mnt/a.test.file
# ls /mnt/
a.test.file  lost+found
# ls -l /mnt/
total 13
-rw-r--r--    1 root     root            1 Apr  2 00:28 a.test.file
drwxr-xr-x    2 root     root        12288 Apr  2 00:28 lost+found
```

Ok, we now have something to work with. Let's make the snapshot:

```
# lvcreate --size 16m --snapshot --name snap /dev/test/HOWTO
lvcreate -- WARNING: all snapshots will be disabled if more than 16 MB are changed
```

```
lvcreate -- INFO: using default snapshot chunk size of 64 KB
lvcreate -- doing automatic backup of "test"
lvcreate -- logical volume "/dev/test/HOWTO" successfully created
```

More on the '−−size' parameter later. Let's mount the snapshot:

```
# mount /dev/test/snap /snap
# ls /snap
total 13
-rw-r--r--    1 root     root            1 Apr  2 00:28 a.test.file
drwxr-xr-x    2 root     root        12288 Apr  2 00:28 lost+found
```

Now we erase a.test.file from the original, and check if it's still there in the snapshot:

```
# rm /mnt/a.test.file
# ls /snap
total 13
-rw-r--r--    1 root     root            1 Apr  2 00:28 a.test.file
drwxr-xr-x    2 root     root        12288 Apr  2 00:28 lost+found
```

Amazing Mike!

7.1 How does it work?

Remember that we had to set the '−−size' parameter? What really happens is that the 'snap' volume needs to have a copy of all blocks or 'chunks' as LVM calls them, which are changed in the original.

When we erased a.test.file, it's inode was removed. This caused 64 KB to be marked as 'dirty' − and a copy of the original data was written to the 'snap' volume. In this case we allocated 16MB for the snapshot, so if more than 16MB of "chunks" are modified, the snapshot will be deactivated.

To determine the correct size for a snapshot partition, you will have to guess based on usage patterns of the primary LV, and the amount of time the snapshot will be active. For example, an hour−long backup in the middle of the night when nobody is using the system may require very little space.

Please note that snapshots are not persistent. If you unload LVM or reboot, they are gone, and need to be recreated.

8.Redundancy & Performance

For performance reasons, it is possible to spread data in a 'stripe' over multiple disks. This means that block 1 is on Physical Volume A, and block 2 is on PV B, while block 3 may be on PV A again. You can also stripe over more than 2 disks.

This arrangement means that your have more disk bandwidth available. It also means that more 'spindles' are involved. More on this later.

Besides increasing performance, it is also possible to have your data in copies on multiple disks. This is called mirroring. Currently, LVM does not support this natively but there are ways to achieve this.

8.1 Why stripe?

Disk performance is influenced by three things, at least. The most obvious is the speed at which data on a disk can be read or written sequentially. This is the limiting factor when reading or writing a large file on a SCSI/IDE bus with only a single disk on it.

Then there is the bandwidth available TO the disk. If you have 7 disks on a SCSI bus, this may well be less than the writing speed of your disk itself. If you spend enough money, you can prevent this bottleneck from being a problem.

Then there is the latency. As the saying goes, latency is always bad news. And even worse, you can't spend more money to get lower latency! Most disks these days appear to have a latency somewhere around 7ms. Then there is the SCSI latency, which used to be something like 25ms.

FIXME: need recent numbers!

What does this mean? The combined latency would be around 30ms in a typical case. You can therefore perform only around 33 disk operations per second. If you want to be able to do many thousands of queries per second, and you don't have a massive cache, you are very much out of luck.

If you have multiple disks, or 'spindles', working in parallel, you can have multiple commands being performed concurrently, which nicely circumvents your latency problem. Some applications, like a huge news server, don't even work anymore without striping or other IO smartness.

This is what striping does. And, if your bus is up to it, even sequential reading and writing may go faster.

8.2 Why not

Striping without further measures raises your fault chance, on a 'per bit' basis. If any of your disks dies, your entire Logical Volume is gone. If you just concatenate data, only part of your filesystem is gone.

The ultimate option is the mirrored stripe.

FIXME: make a mirrored stripe with LVM and md

8.3 LVM native striping

Specifying stripe configuration is done when creating the Logical Volume with lvcreate. There are two relevant parameters. With −i we tell LVM how many Physical Volumes it should use to scatter on. Striping is not really done on a bit−by−bit basis, but on blocks. With −I we can specify the granulation in kilobytes.

Note that this must be a power of 2, and that the coarsest granulation is 128Kbyte.

Example:

```
# lvcreate -n stripedlv -i 2 -I 64 mygroup -L 20M
lvcreate -- rounding 20480 KB to stripe boundary size 24576 KB / 6 PE
lvcreate -- doing automatic backup of "mygroup"
lvcreate -- logical volume "/dev/mygroup/stripedlv" successfully created
```

Performance notices

The performance 'gain' may well be very negative if you stripe over 2 partitions of the same disk – take care to prevent that. Striping with two disks on a single IDE bus also appears useless – unless IDE has improved beyond what I remember.

FIXME: is this still true?

Older motherboards may have two IDE buses, but the second one used to be castrated, dedicated to serving a slow cdrom drive. You can perform benchmarks with several tools, the most noteworthy being 'Bonnie'. The ReiserFS people have released Bonnie++ which may be used to measure performance data.

8.4 Hardware RAID

Many high end Intel x86 servers have Hardware RAID controlers. Most of them have atleast 2 independant SCSI channels. Fortunatly, his has very little bearing on LVM. Before Linux can see anything on such a controler the administrator must define a Logical drive within the raid controler itself. As an example [s]he could choose to stripe together two disks on SCSI channel A and then mirror them onto two disks on channel B. This is a typical RAID 0/1 configuration that maximises performance and data security. When Linux boots on a machine with this configuration it can only 'see' one disk on the RAID controler and that is the Logical drive that contains the four disks in the RAID 0/1 stripeset. This means, as far as LVM is concerned, that there is just one disk in the machine and it is to be used as such. If one of the disks fails, LVM wont even know. When the administrator replaces the disk (even on the fly with HotSwap hardware) LVM wont know about that either and the controler will resync the mirrored data and all will be well. This is where most people take a step back and ask "Then what good does LVM do for me with this RAID controler?" The easy answer is, in most cases, after you define a logical drive in the RAID controler you cant add more disks to that drive later. So if you miscalculate the space requirements or you simply need more space you cant add a new disk or set of disks into a pre–exsisting stripeset. This means you must create a new RAID stripset in the controler and then with LVM you can simply extend the LVM Logical volume so that it seamlessly spans both stripesets in the RAID controler.

FIXME: Is there more needed on this subject ?

8.5 Linux software RAID

Linux 2.4 comes with very good RAID in place. Linux 2.2 by default, as released by Alan Cox, features an earlier RAID version that's not well regarded. The reason that 2.2 still features this earlier release is the the kernel people don't want to make changes within a stable version that require userland updates.

Most people, which included Red Hat, Mandrake and SuSE, chose to replace it with the 0.90 version which appears to be excellent.

We will only treat the 0.90 version here.

FIXME: write more of this

9.Cookbook

9.1 Moving LVM disks between computers

With all this new technology, simple tasks like moving disks from one machine to another can get a bit tricky. Before LVM users only had to put the disk into the new machine and mount the filesystems. With LVM there is a bit more to it. The LVM structures are saved both on the disks and in the /etc/lvmconf directory so the only thing that has to be done to move a disk or a set of disks that contain a Volume Group is to make sure the machine that the VG belonged to will not miss it. That is accomplished with the `vgexport` command. `vgexport` simply removes the structures for the VG from /etc/lvmconf, but does not change anything on the disks. Once the disks are in the new machine (they don't have to have the same ID's) the only thing that has to be done is to update /etc/lvmconf. Thats done with `vgimport`.

Example:

On machine #1:

```
vgchange -a n vg01
vgexport vg01
```

On machine #2:

```
vgimport vg01 /dev/sda1 /dev/sdb1
vgchange -a y vg01
```

Notice that you don't have to use the same name for the Volume Group. If the vgimport command did not save a configuration backup use `vgcfgbackup` to do it.

9.2 Rebuilding /etc/lvmtab and /etc/lvmtab.d

FIXME: write about more neat stuff

10.Further reading

LVM site

The main LVM resource available

German LVM HOWTO

If you can read German, this already contains a lot of information

Translation of the German HOWTO

Peter.Wuestefeld@resnova.de is translating the German HOWTO into English. It appears that they will soon be investing lots of time in it. If you doubt our HOWTO or miss something, please try their effort.

HP/UX Managing Disks Guide

Since the Linux LVM is almost an exact workalike of the HP/UX implementation, their documentation is very useful to us as well. Very good stuff.

11.Acknowledgements & Thanks to

We try to list everybody here who helped make this HOWTO. This includes people who send in updates, fixes or contributions, but also people who have aided our understanding of the subject.

- Axel Boldt <axel@uni−paderborn.de>
- Sean Reifschneider <jafo@tummy.com>
- Alexander Talos <at@atat.at>
- Eric Maryniak <e.maryniak@pobox.com>
