MindWorks UK

LVM2, in English

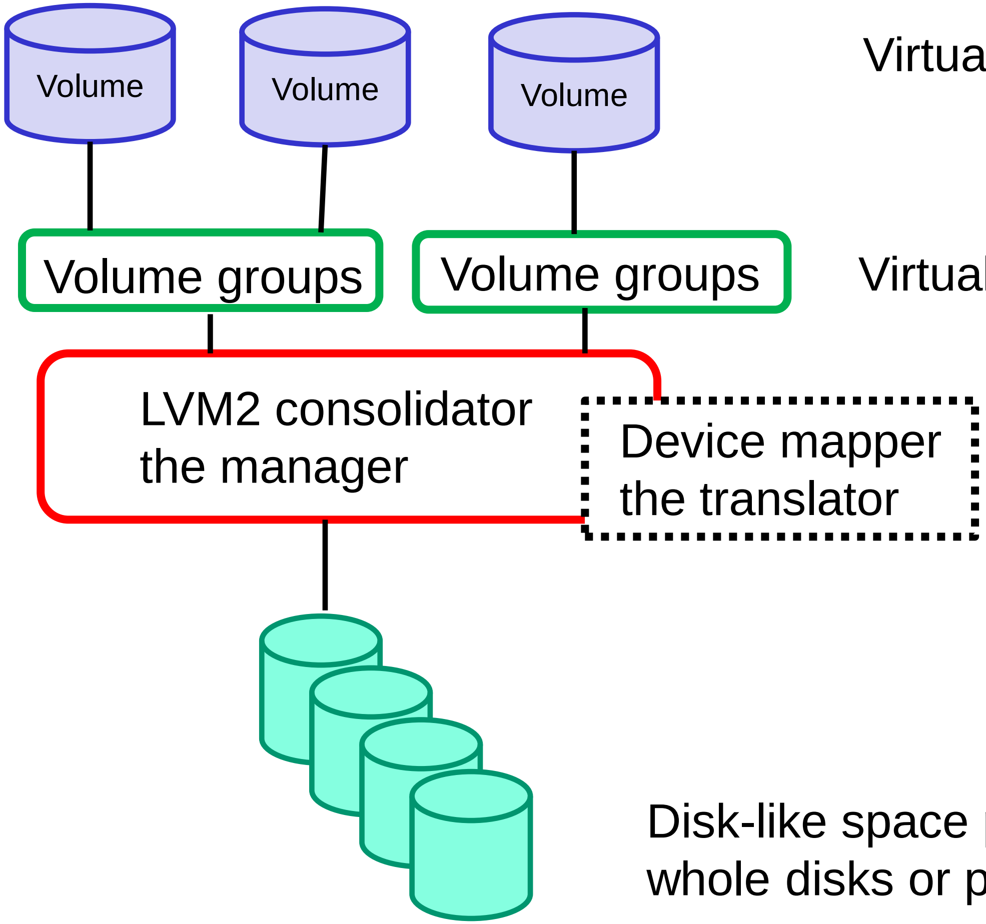
Joe Doupnik

Univ of Oxford, MindworksUK

jrd@netlab1.oucs.ox.ac.uk

Topology picture, it helps

File systems live up here, on each partition



Virtual disk-like partitions

Virtual whole disk-like devices

Disk-like space providers:
whole disks or partitions

Teminology, a confusing part

LVM2-speak	Ordinary usage
On the top side (where file systems live):	
Volume	**disk partition**
Volume Group	**whole disk**
	VG is equivalent to NSS Pool
On the bottom side (where disks live):	
Physical volume	**whole disk or partition** **or what masquerades as them**

Planting the LVM2 flag

LVM2 acquires physical volumes (space) by owning either partitions and/or whole disks

It stamps each partition with type 0x83, other partitions are unaffected

It steals a tiny amount of space on each partition or disk to write its bookkeeping of which volume group and volume it belongs to

Such space has only one owner (not shared)

The LVM2 staff

LVM2 is a manager, a consolidator of space, but mostly a manager with user-space controls (*lvm*)

Under the covers lurks *device-manager* (DM) whose task it is to convert a “volume’s” block number (that seen by the file system) to the correct physical disk and block number on it

DM does table lookups, quickly

DM can be used on its own as well (*dmsetup*)

LVM2 job description details

Once LVM2 has its hands on a disk block from above it can do many things with it:

- **Ask DM to find the matching storage spot**
- **Duplicate the block to multiple devices (RAID)**
- **Switch amongst storage devices (Stripes)**
- **Pass it to the volume encryption module**
- **Use alternative communication paths (multipath)**
- **And other activities such as create “snapshots”**

Device mapper does mapping and other drudgery on behalf of LVM2

Device mapper user controls, yikes!

```
NAME
     dmsetup - low level logical volume management

SYNOPSIS
dmsetup help [-c|-C|--columns]
dmsetup create device_name [-u uuid] [--notable | --table <table> | table_file]
dmsetup remove [-f|--force] device_name
dmsetup remove_all [-f|--force]
dmsetup suspend [--nolockfs] [--noflush] device_name
dmsetup resume device_name
dmsetup load device_name [--table <table> | table_file]
dmsetup clear device_name
dmsetup reload device_name [--table <table> | table_file]
dmsetup rename device_name new_name
dmsetup message device_name sector message
dmsetup ls [--target target_type] [--exec command] [--tree [-o options]]
dmsetup info [device_name]
dmsetup info -c|-C|--columns [--noheadings] [--separator separator] [-o fields] [-O|--sort sort_fields] [device_name]
dmsetup export [device_name]
dmsetup deps [device_name]
dmsetup status [--target target_type] [device_name]
dmsetup table [--target target_type] [--showkeys] [device_name]
dmsetup wait device_name [event_nr]
dmsetup mknodes [device_name]
dmsetup udevcreatecookie
dmsetup udevreleasecookie [cookie]
dmsetup udevflags cookie
dmsetup udevcomplete cookie
dmsetup udevcomplete_all
dmsetup udevcookies
dmsetup targets
dmsetup version
dmsetup setgeometry device_name cyl head sect start
dmsetup splitname device_name [subsystem]

devmap_name major minor
devmap_name major:minor
```

Table is “start block” “end block” “linear or striped” “device” “start block on device”

LVM commands

dumpconfig	Dump active configuration
formats	List available metadata formats
help	Display help for commands
lvchange	Change the attributes of logical volume(s)
lvconvert	Change logical volume layout
lvcreate	Create a logical volume
lvdisplay	Display information about a logical volume
lvextend	Add space to a logical volume
lvmchange	With the device mapper, this is obsolete and does nothing.
lvmdiskscan	List devices that may be used as physical volumes
lvmsadc	Collect activity data
lvmsar	Create activity report
lvreduce	Reduce the size of a logical volume
lvremove	Remove logical volume(s) from the system
lvrename	Rename a logical volume
lvresize	Resize a logical volume
lvs	Display information about logical volumes
lvscan	List all logical volumes in all volume groups

LVM commands

pvchange	Change attributes of physical volume(s)
pvresize	Resize physical volume(s)
pvck	Check the consistency of physical volume(s)
pvcreate	Initialize physical volume(s) for use by LVM
pvdata	Display the on-disk metadata for physical volume(s)
pvdisplay	Display various attributes of physical volume(s)
pvmove	Move extents from one physical volume to another
pvremove	Remove LVM label(s) from physical volume(s)
pvs	Display information about physical volumes
pvscan	List all physical volumes
segtypes	List available segment types

vgcfgbackup	Backup volume group configuration(s)
vgcfgrestore	Restore volume group configuration
vgchange	Change volume group attributes
vgck	Check the consistency of volume group(s)

LVM commands

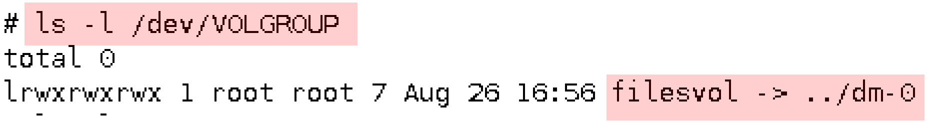
vgconvert	Change volume group metadata format
vgcreate	Create a volume group
vgdisplay	Display volume group information
vgexport	Unregister volume group(s) from the system
vgextend	Add physical volumes to a volume group
vgimport	Register exported volume group with system
vgmerge	Merge volume groups
vgmknodes	Create the special files for volume group devices in /dev
vgreduce	Remove physical volume(s) from a volume group
vgremove	Remove volume group(s)
vgrename	Rename a volume group
vgs	Display information about volume groups
vgscan	Search for all volume groups
vgsplit	Move physical volumes into a new or existing volume group
version	Display software and driver version information

Yes, these repeat many done by dmsetup

Sundry /dev names

LVM2 creates directories such as *\/dev/volgroup*

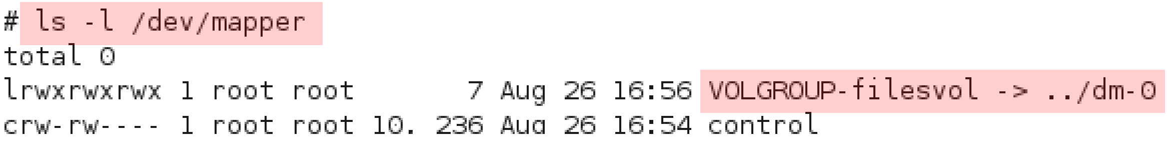
Within an LVM2 directory are names of "volumes" as symbolic links to DM devices *\/dev/dm-digit*



```
# ls -l /dev/VOLGROUP
total 0
lrwxrwxrwx 1 root root 7 Aug 26 16:56 filesvol -> ../dm-0
```

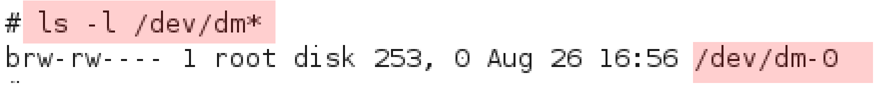
"volume" filesvol

DM creates device names as *\/dev/mapper/volgroup-volume* & similar, all under *\/dev/mapper*



```
# ls -l /dev/mapper
total 0
lrwxrwxrwx 1 root root       7 Aug 26 16:56 VOLGROUP-filesvol -> ../dm-0
crw-rw---- 1 root root 10, 236 Aug 26 16:54 control
```

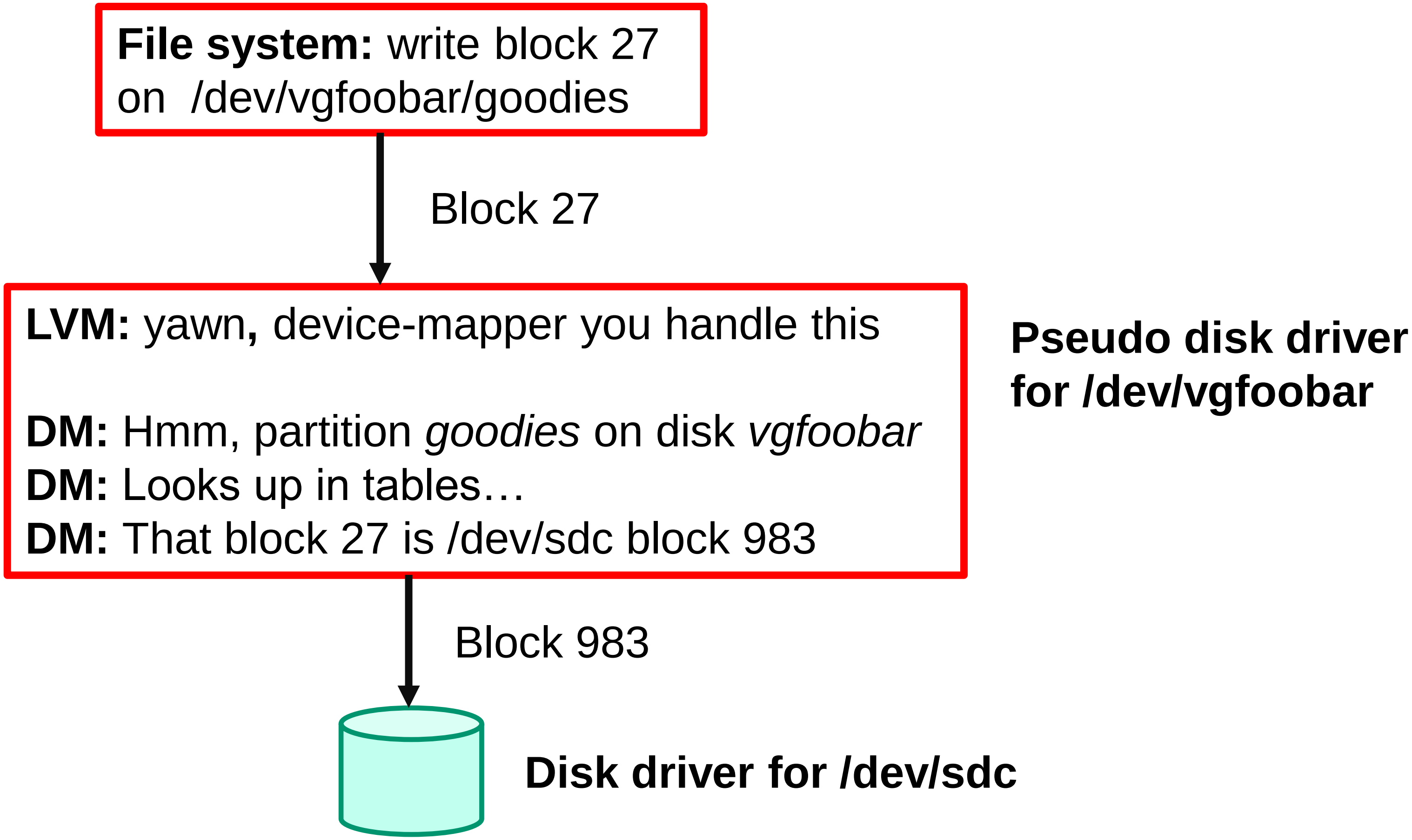
These names are symbolic links to *\/dev/dm-digit*



```
# ls -l /dev/dm*
brw-rw---- 1 root disk 253, 0 Aug 26 16:56 /dev/dm-0
```

DM devices (*\/dev/dm-digit*) use major number 253

Normal work: just a table lookup



File system: write block 27 on /dev/vgfoobar/goodies

Block 27

LVM: yawn, device-mapper you handle this

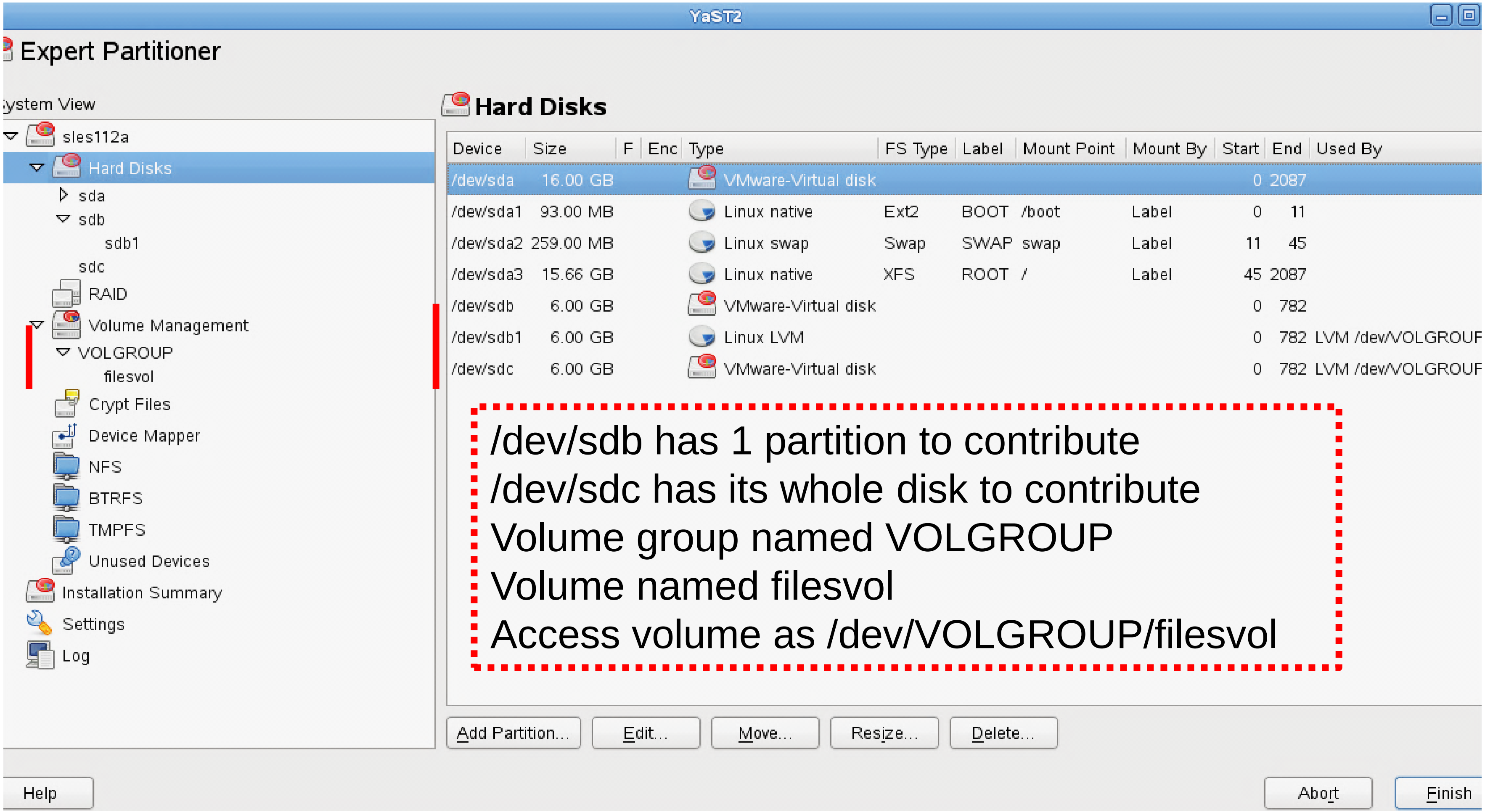
DM: Hmm, partition *goodies* on disk *vgfoobar*
DM: Looks up in tables…
DM: That block 27 is /dev/sdc block 983

Pseudo disk driver for /dev/vgfoobar

Block 983

Disk driver for /dev/sdc

Partitioner view of a system



Device	Size	F	Enc	Type	FS Type	Label	Mount Point	Mount By	Start	End	Used By
/dev/sda	16.00 GB			VMware-Virtual disk					0	2087	
/dev/sda1	93.00 MB			Linux native	Ext2	BOOT	/boot	Label	0	11	
/dev/sda2	259.00 MB			Linux swap	Swap	SWAP	swap	Label	11	45	
/dev/sda3	15.66 GB			Linux native	XFS	ROOT	/	Label	45	2087	
/dev/sdb	6.00 GB			VMware-Virtual disk					0	782	
/dev/sdb1	6.00 GB			Linux LVM					0	782	LVM /dev/VOLGROUF
/dev/sdc	6.00 GB			VMware-Virtual disk					0	782	LVM /dev/VOLGROUF

/dev/sdb has 1 partition to contribute
/dev/sdc has its whole disk to contribute
Volume group named VOLGROUP
Volume named filesvol
Access volume as /dev/VOLGROUP/filesvol

Not all "disks" are really "disks"

Kernel modules can proclaim to be disk handlers (just fill in the proper form and submit it to the kernel)

Such pseudo-disks include: loop driver, ramdisk, DRBD driver, encryption, LVM2 stuff, and more

Thus we can have a stack, a layer cake, of various flavours of "disk", some contribute, others consume

LVM2 does the block-in/block-out mapping

Where is information kept?

On disk, in the private working space on an acquired disk or partition

Be wary of booting to an LVM disk: best to have /boot as a regular partition to not confuse poor grub

Grub2 is finally able to recognize LVM volumes

NLVM vs LVM2

Novell’s NLVM (new with OES11) is similar to LVM but uses its own data structures

Ownership is via partition id 0x65 (NetWare)

Device-mapper is used to define the space

LVM2 is not involved, EVMS has gone away

Resizing VG’s and volumes

We can tinker with LVM2’s mapping tables, and its set of disk space providers

A VG (aka disk) can be expanded or shrunk, but volumes will be unaware of that (provided their space still exists)

A volume (aka partition) may be expanded or shrunk, but the owning file system needs to be informed (some can change, others not)

Changing file system size is dangerous, beware

Snapshots work on logical volumes

```
lvconvert [-s|--snapshot]
        [-c|--chunksize]
        [-d|--debug]
        [-h|-?|--help]
        [--noudevsync]
        [-v|--verbose]
        [-Z|--zero {y|n}]
        [--version]
        OriginalLogicalVolume[Path]  SnapshotLogicalVolume[Path]

lvconvert --merge
        [-b|--background]
        [-i|--interval seconds]
        [-d|--debug]
        [-h|-?|--help]
        [-v|--verbose]
        SnapshotLogicalVolume[Path]
```

lvcreate can also create snapshot volumes

Snapshots cont’d

Using **lvconvert**, helped by device mapper (user level control by dmsetup)

-s, --snapshot

Create a snapshot from existing logical volume using another existing logical volume as its origin.

--merge

Merges a snapshot into its origin volume. When merging starts, the resulting logical volume will have the origin's name, minor number and UUID.

While the merge is in progress, reads or writes to the origin appear as they were directed to the snapshot being merged. When the merge finishes, the merged snapshot is removed. Multiple snapshots may be specified on the command line or a @tag may be used to specify multiple snapshots be merged to their respective origin.

Merging copies the snapshot volume onto the original volume, thus changing a snapshot can eventually be copied onto the original.

Best to think of merge as the snapshot volume assumes the name of the original.

Snapshots are problematic with LVM2. They are easy to create but not easy to remove.

Snapshot DM devices

When you create the first LVM2 snapshot of a volume, <u>four dm devices are used</u>:

1) **a device (-real) containing the original mapping table of the source volume**
2) **a device (-cow) used as the <COW device>**
3) **a "snapshot" device, combining #1 and #2, which is the visible snapshot volume**
4) **the "original" volume (which uses the device number used by the original source volume), whose table is replaced by a "snapshot-origin" mapping from device #1.**

A fixed naming scheme is used, so with the following commands:

```
lvcreate -L 1G -n base volumeGroup
lvcreate -L 100M --snapshot -n snap volumeGroup/base
```

we'll have this situation (with volumes in above order):

```
# dmsetup table | grep volumeGroup
        volumeGroup-base-real:  0 2097152 linear 8:19 384
        volumeGroup-snap-cow: 0   204800 linear 8:19 2097536
        volumeGroup-snap: 0 2097152 snapshot 254:11 254:12 P 16
        volumeGroup-base: 0 2097152 snapshot-origin 254:11
# ls -lL /dev/mapper/volumeGroup-*
        brw------- 1 root root 254, 11 29 ago 18:15 /dev/mapper/volumeGroup-base-real
        brw------- 1 root root 254, 12 29 ago 18:15 /dev/mapper/volumeGroup-snap-cow
        brw------- 1 root root 254, 13 29 ago 18:15 /dev/mapper/volumeGroup-snap
        brw------- 1 root root 254, 10 29 ago 18:14 /dev/mapper/volumeGroup-base
```

http://www.kernel.org/doc/Documentation/device-mapper/snapshot.txt

Snapshot example

Use /dev/sdb and /dev/sdc each is a 16GB disk

Create a 32GB pseudo disk (VG) named VGROUP

```
vgcreate VGROUP  /dev/sdb  /dev/sdc
```

Create a 16GB pseudo partition (Volume) named vol1

```
lvcreate –L 16G –n vol1 VGROUP
```

This appears as /dev/VGROUP/vol1

Create a file system on vol1 and mount it

```
mkfs.xfs –L VOL1 /dev/VGROUP/vol1                -L for label
mount –o noatime,nodiratime  /dev/VGROUP/vol1 /home/XFS
```

Snapshot example

Create snapshot of vol1, named vol1snap. Sized to hold most saved original disk blocks (beware overflows)

```
lvcreate –s –L 15G --name vol1snap  /dev/VGROUP/vol1
```

Yields /dev/VGROUP/vol1 **and** /dev/VGROUP/vol1snap

Mount the snapshot, -o nouuid is for XFS which has unique identifiers for each file system

```
mount –o nouuid  /dev/VGROUP/vol1snap  /mnt
```

Removing a snapshot , ugh

umount **where snapshot may be mounted, then**

```
lvremove –f /dev/VGROUP/vol1snap
```

These commands may be needed to cleanup:

```
umount /dev/VGROUP/vol1
dmsetup remove /dev/mapper/VGROUP-vol1snap-cow
dmsetup remove /dev/mapper/VGROUP-vol1snap
lvchange –an /dev/VGROUP/vol1                 disable vol1
lvchange –ay /dev/VGROUP/vol1                 enable vol1
mount –o options  /dev/VGROUP/vol1  /place
```

Making errors here is easy, we can lose everything

Insightful comments on LVM2 and snapshots

http://www.linuxjournal.com/article/1090

http://serverfault.com/questions/279571/lvm-dangers-and-caveats

Read the whole thing, including the useful comments by others

My opinion on LVM snapshots: overly complicated, poor LVM vs DM coordination, space allocation math, difficult to remove later

Loss of a physical volume

LVM2 is able to continue operating with a missing pv, though references to it will result in errors

This is a mixed blessing as LVM2 may remember what we had removed permanently. Eradicating such memory takes more commands, with risks.

Performance

Saved original disk blocks are written to a separate device, a Copy On Write procedure

The original can change, but the snapshot observes the original disk blocks

Two side effects of this:

1. Write throughput drops dramatically when an old block needs to be copied
2. COW writes are put on save disk in order of appearance, not where they would be on the whole file system, which leads to scattered accesses

Non-copy writes are at nearly native disk speed

A moving target

Device mapper is undergoing improvements:

- thin provisioning (sparse files, a la VMware)
- snapshots, but still complicated
- disk block replication

For proper disk block replication between machines look at DRBD, particularly version 9 which is in development. Version 8.4.1 is in SLES 11 SP2 HA

BTRFS: filesystem+LVM all in one

BTRFS combines its file system with its own volume management

BTRFS can expand or shrink its space usage, combining underlying space providers

BTRFS supports snapshots and detachable file systems (aka “subvolumes”)

BTRFS user level commands

```
NAME
      btrfs - control a btrfs filesystem

SYNOPSIS
      btrfs device add <dev> [<dev>..] <path>
      btrfs device delete <dev> [<dev>..] <path>
      btrfs device scan [--all-devices|<device> [<device>...]
              …
      btrfs subvolume create [<dest>/]<name>
      btrfs subvolume delete <subvolume>
      btrfs subvolume find-new <subvolume> <last_gen>
      btrfs subvolume list [-p] <path>
      btrfs subvolume set-default <id> <path>
      btrfs subvolume get-default <path>
      btrfs subvolume snapshot [-r] <source> [<dest>/]<name>
```

Create a writable or readonly snapshot of the subvolume <source> with the name <name> in the <dest> directory. All parts use BTRFS.

Manage with “snapper” by SUSE

Tools

YaST *Partitioner* is a useful simple tool, but covers only a small portion of possibilities

lvm, the cover shell over LVM and Device Mapper

dmsetup, the shell controlling Device Mapper

nssmu for NLVM

My advice

Aggregate space if you must, but otherwise avoid

Don’t depend on resizing file systems later; the process is dangerous and not all f/s will do it

LVM snapshots are a muddle, best avoided if possible

Keep in mind its simple task of renumbering