

LVM on SW RAID for sarge – success

Source: <http://linux.derkeiler.com/Mailing-Lists/Debian/2004-07/2069.html>

From: George Karaolides (george_at_otenet-telecom.com)

Date: 07/14/04

To: debian-boot@lists.debian.org, debian-user@lists.debian.org

Date: Wed, 14 Jul 2004 10:49:32 +0300

Greetings,

I've just had success migrating Debian Sarge to root on LVM on RAID. Here's an account of what I've done. I believe it could be of interest to both the debian–installer people on debian–boot and to the Debian user community on debian–user, hence the cross–posting to both lists; apologies to anyone subscribed to both. I'm not subscribed to either, so any replies please CC me.

My Sarge Root–onLVM effort improves on my Woody effort:

<http://karaolides.com/computing/HOWTO/lvmraid/>

and on Massimiliano Ferrero's effort:

<http://www.midhgard.it/docs/lvm/html/>

in that no kernel recompiling, or composition of config. files, is required; all is achieved with the Debian Sarge software distribution and a few commands.

Hardware: twin Intel processors, 3x SCSI SCA disks

I proceeded as follows:

– Install Sarge on the first disk

First make sure the SCSI BIOS boots from the first disk.

I did a net–install off the Internet using the boot, root and net–drivers floppies from last Monday's (20041207) daily builds. The disk was partitioned to reflect the intended final filesystem structure; for a server I usually make:

```
/
/boot
swap
```

Debian–User: LVM on SW RAID for sarge – success

/tmp
/usr
/usr/local
/var
/var/log
/home

As this disk is only to be used for initial installation and will later be incorporated into the RAID arrays, 256MB partitions are more than adequate.

I made reiserfs on all the filesystem partitions and completed the installation normally. I installed kernel 2.4.26–smp.

I have stuck with reiserfs. I wanted to use XFS so I could get quota support without patches, but tests indicated that XFS is not as reliable as reiserfs. My "torture test" goes like: create an LV, make filesystem on it, mount it, copy the Linux bziped source tarball into it, start extracting, and while extracting do an LV extend followed by a filesystem resize, as usually done to increase filesystem space on a live system. Reiserfs has been passing this test since at least late 2002. Tried it on Sarge twice with XFS, the filesystem was trashed both times.

Set up devfs, devfsd and install lvm10

Having completed installation and base–config, I then edited /boot/grub/menu.lst and added the "devfs=mount" kernel boot option, as required by lvm10 (if using a kernel compiled with devfs support with lvm10, devfs must be mounted on /dev, as stated in the warning issued when installing the Debian lvm10 package). Then of course I ran grub–install to update the MBR with the new boot option. I also installed lvm10 and dependencies, and devfsd. Then I rebooted to check that devfs was mounted and devfsd was running.

Set up RAID:

The remaining two disks were partitioned and used to create RAID arrays in degraded mode, into which the first disk would eventually be incorporated. I usually make two arrays:

1. RAID1 made up of 64MB partiotions at the beginning of each disk, for /boot
2. RAID5 made up of partitions comprising the remaining space on each disk, to be used for LVM.

I had some problems with mdadm not initialising my arrays properly. I eventually found that for some reason, when the second partition on each disk (comprising all the space remaining after the 64M partition at the beginning of each disk was made) was made as a primary partition, and

Debian–User: LVM on SW RAID for sarge – success

then mdadm used to assemble it into a RAID array, a RAID superblock would be written to both the partition device and the disk device, as evidenced by the output of e.g.

```
mdadm --examine /dev/sdb  
mdadm --examine /dev/sdb2
```

both showing a RAID superblock.

This caused strange things to happen when mdadm tried to initialise arrays on reboot, e.g. unuseable RAID arrays comprising of disks (rather than partitions) appearing, or disk devices appearing together with partition devices in an array and the array names (/dev/md0, /dev/md1) appearing in a different order to that with which they were created etc.

I discovered that making the 64MB partitions for the RAID1 elements as primary partitions and the second partitions for the RAID5 elements as logical partitions allowed mdadm to assemble the arrays properly on reboot.

Create the degraded RAID arrays:

```
mdadm -C -11 -n3 /dev/md0 missing /dev/sdb1 /dev/sdc1  
mdadm -C -15 -n3 /dev/md1 missing /dev/sdb5 /dev/sdc5
```

Note /dev/sd*5 for the second array, as these are logical partitions and numbering starts from 5

Then run

```
dpkg-reconfigure mdadm
```

and make sure the option to autostart RAID arrays at boot is selected.

Use the RAID1 for /boot, and the RAID5 for LVM:

```
mkfs.reiserfs /dev/md0  
pvcreate /dev/md/1 (note devfs path used as per lvm10 warning)
```

– Create volume group and volumes:

```
vgcreate vg0 /dev/md/1  
lvcreate -n root -L256M vg0  
lvcreate -n tmp -L256M vg0  
lvcreate -n swap -L1024 vg0  
lvcreate -n var -L256M vg0  
lvcreate -n varlog -L256M vg0  
lvcreate -n usr -L256M vg0  
lvcreate -n usrlocal -L256M vg0
```

etc.

LVM on SW RAID for sarge – success

Debian–User: LVM on SW RAID for sarge – success

Then format with reiserfs:

```
mkreiserfs /dev/vg0/root  
etc.
```

Make swap space:

```
mkswap /dev/vg0/swap
```

Mount root LV:

```
mount /dev/vg0/root /mnt
```

Copy over root fs:

```
cp -avx / /mnt
```

Mount the RAID1 for /boot and copy over /boot:

```
mount /dev/md0 /mnt/boot  
cp -avx /boot /mnt
```

Mount the /home, /usr and /var LV's and copy over filesystems:

```
mount /dev/vg0/home /mnt/home  
mount /dev/vg0/var /mnt/usr  
mount /dev/vg0/usr /mnt/var  
cp -avx /home /usr /var /mnt
```

Then mount the /var/log and /usr/local LV's:

```
mount /dev/vg0/usrlocal /mnt/usr/local  
mount /dev/vg0/varlog /mnt/var/log
```

Copy over /usr/local and /var/log:

```
cp -avx /usr/local /mnt/usr  
cp -avx /var/log /mnt/var
```

The system is now copied over to LVM–on–RAID. It should be made bootable. Before this can be done, /dev and /proc should be available when chrooting into the target system:

```
mount -t devfs devfs /mnt/dev  
mount -t proc proc /mnt/proc
```

Edit /mnt/etc/fstab to reflect the filesystem structure, replacing /dev/sda* with /dev/vg0/* (not forgetting /dev/md0 for /boot)

Make a new LVM–on–RAID capable initrd image in the target system: mkinitrd is clever enough to figure out what's needed from the new fstab

Debian–User: LVM on SW RAID for sarge – success

file.

```
chroot /mnt mkinitrd -o /boot/<name-of-existing-initrd-image>
```

Edit /mnt/boot/grub/menu.lst and change the kernel boot option for the root filesystem from /dev/sda* to /dev/vg0/root. The line in my file looked like:

```
kernel /vmlinuz-2.4.26-1-686-smp root=/dev/vg0/root devfs=mount noapic ro single
```

(I gave it noapic because of APIC problems with my motherboard killing the ethernet card, you probably don't care about that)

Then run grub in the target system:

```
chroot /mnt grub
```

On the grub command line, tell grub to use one of the partitions in the RAID1 for /boot to read data, e.g. second disk, first partition:

```
root (hd1,0)
```

Then tell it to write an MBR on the second disk:

```
setup (hd1)
```

And also on the third disk:

```
setup (hd2)
quit
```

It should now be possible to boot into the initial single-disk system when booting from the first disk, and into the degraded LVM-on-RAID system when booting from the second or third disks; use the SCSI BIOS to select.

Once it is verified that the LVM-on-RAID system boots and functions correctly, the first disk can be repartitioned like the second and third ones and the partitions incorporated into the RAID arrays:

```
mdadm --add /dev/md0 /dev/sda1
mdadm --add /dev/md1 /dev/sda5
```

Make the system also bootable from the first disk:

```
grub
root (hd1,0)
setup (hd0)
quit
```

Debian-User: LVM on SW RAID for sarge – success

Treat yourself to your favourite beverage while watching the array reconstruction :)

```
watch -n3 cat /proc/mdstat
```

Enjoy redundant, high-performance, instantly-online-resizeable filesystems:

```
lvextend -L+<how-much-more-space-do-you-want>[MG] /dev/vg0/<filesystem>;  
resize-reiserfs /dev/vg0/<filesystem>
```

Snapshots:

Note that snapshots of journalled filesystems like reiserfs do not work unless a kernel patch is applied and the kernel recompiled. It is possible to generate the patch by downloading the Debian package of the kernel source, and the Debian source package for lvm10. Anyone interested in snapshots drop me a line and I'll tell you how to do it.

--

Best regards,
George Karaolides
System Administrator
OTEnet Telecom
20 Ayias Paraskevis St.
CY-2002 Strovolos, Nicosia, Cyprus
tel: +357 22 693333
fax: +357 22 455686
www.otenet-telecom.com
Confidentiality notice and disclaimer applies:
<http://www.otenet-telecom.com/Disclaimer.html>

--

To UNSUBSCRIBE, email to debian-user-REQUEST@lists.debian.org
with a subject of "unsubscribe". Trouble? Contact listmaster@lists.debian.org