Upgrade your system to software RAID 1/4/5

Alessandro Dotti Contra
alessandro@hyboria.org

April 25, 2010

Abstract

This document is intended to be a dry step by step guide to upgrade an existing Linux non-raid system to RAID 1/4/5. The explanation will focus on RAID 1 (mirror) for the system disks, and RAID 5 for the home directories.

Prepare the hardware

Specific hardware configuration may be considered or required only if you need to upgrade an IDE based system. A general rule suggests to use only one IDE device for each IDE channel available; the device *must* be set as the master device for the channel. If you need to use a CD-ROM or some other IDE device, consider to use a supplemental IDE controller.

SCSI systems do not have specific hardware configuration requirements, thought it may be desiderable to apply some specific configuration related to the actual SCSI subsystem available.

Prepare a new kernel

The first thing to do is to prepare a new kernel which supports md devices (multiple devices).

RAID support must be enable by selecting the appropriate options in the kernel configuration menu (you can choose multiple RAID personalities if you wish, of course).

```
Multi-device support (RAID and LVM):
        [*] Multiple devices driver support (RAID and LVM)
                <*>  RAID support
                < >   Linear (append) mode
                < >   RAID-0 (striping) mode
                <*>   RAID-1 (mirroring) mode
                <*>   RAID-4/RAID-5 mode
```

Now boot the system with your brand new kernel; to be sure the RAID personalities your going to use are correctly supported you, can examine the `/proc/mdstat` file.

At this stage you also need to install `mdadm` to be able to manage your raid devices when needed.

Configure your brand new RAID

For ease of discussion, we will consider from now on `/dev/hda` as the *old* system disk, and `/dev/hdc` as the *new* one. For the home array, `/dev/sda` will be the *old* disk, while `/dev/sdb` and `/dev/sdc` will be the *new* ones, and `/dev/sdd` the spare disk we're going to use.

We will consider the following partitions layout:

`/dev/hda1`: `/boot` partition

`/dev/hda2`: `swap` partition

`/dev/hda3`: `/` partition

`/dev/sda1`: `/home` partition

Partition the hard drives

The first thing to do is to clone the partition table from `/dev/hda` to `/dev/hdc`. Your new disks *must* be bigger or equal, in size, than the old ones.

Proceed as follow:

```
$ sfdisk -d /dev/hda > ptable.hda
$ sfdisk /dev/hdc < ptable.hda
```

Follow the same steps for `/dev/sdb`, `/dev/sdc` and `/dev/sdd`:

```
$ sfdisk -d /dev/sda > ptable.sda
$ sfdisk /dev/sdb < ptable.sda
$ sfdisk /dev/sdc < ptable.sda
$ sfdisk /dev/sdd < ptable.sda
```

Now change all the partitions that will be part of the RAID arrays to type *fd* (Linux raid autodetect).

Set a bootable partition

Be sure to have a primary partition with the boot flag set. This is needed by `lilo` to boot correctly from a RAID array.

Create and prepare the RAID devices

Once the hard drives have been successfully partitioned, you have to create the md devices. Do as follows:

```
$ mdadm --create /dev/md0 --level=1 --raid-devices=2 \
    /dev/hdc1 missing
$ mdadm --create /dev/md2 --level=1 --raid-devices=2 \
    /dev/hdc3 missing
$ mdadm --create /dev/md2 --level=5 --raid-devices=3 \
    /dev/sdb1 /dev/sdc1 missing
```

The *missing* parameter used in place of `/dev/hda[13]` (and in place of `/dev/sda1` for the home array) is needed to tell `mdadm` to start the arrays in degraded mode. It *must* be used to leave the *old* disks untouched at this stage; they will be integrated in the arrays later.

If you look at `/proc/mdstat` you should see the RAID devices running.

Now you have to create a file system on each RAID device:

```
$ mkfs.ext3 /dev/md0
$ mkfs.ext3 /dev/md1
$ mkfs.ext3 /dev/md2
```

and finally mount them to recreate the file system layout:

```
$ mount /dev/md1 /mnt
$ mkdir /mnt/boot /mnt/home
$ mkdir /mnt/home /mnt/home
$ mount /dev/md0 /mnt/boot
$ mount /dev/md2 /mnt/home
```

Copy the current OS to the new RAID devices

This is pretty straightforward:

```
$ cd /
$ cp -a /bin /mnt
$ cp -a /boot /mnt/boot
...
$ cp -a /home /mnt/home
$ cp -a /var /mnt
...
```

`/proc` and `/mnt` directories *must not* be copied but rather created:

```
$ mkdir /mnt/proc
$ mkdir /mnt/mnt
```

Other directories might be created as well.

Test your new RAID

First of all edit your `/mnt/etc/fstab` to reflect the new mount points:

```
/dev/md1    /       ext3    errors=remount-ro       0       1
/dev/md0    /boot   ext3    defaults                0       2
/dev/md2    /home   ext3    defaults                0       2
/dev/hda2   none    swap    sw                      0       0
```

then prepare a boot floppy with the new kernel:

```
$ dd if=kernel.image of=/dev/fd0 bs=2k
$ rdev /dev/fd0 /dev/md1
$ rdev -r /dev/fd0 0
$ rdev -R /dev/fd0 1
```

Dismount the RAID devices and reboot the system.

Alternative approach

If you can't, for any reason, use a boot floppy, you can simply reboot the system and type, at the `lilo` prompt:

```
<your kernel image name> root=/dev/md1
```

Integrate the old disk into the RAID array

Repartition the old disks

Turn off the swap file already in use and repartition the old drives to fit them in the new RAID arrays:

```
$ swapoff /dev/hda2
$ sfdisk /dev/hda < ptable.hda
$ sfdisk /dev/sda < ptable.sda
$ swapon /dev/hda2
```

Hot add the old drives

Now you can hot add the old drives to the new RAID arrays:

```
$ mdadm /dev/md0 -a /dev/hda1
$ mdadm /dev/md1 -a /dev/hda3
$ mdadm /dev/md2 -a /dev/sda1
```

If you look at `/proc/mdstat` you can see the RAID devices reconstructing the arrays.

Use multiple swap files

Last step is to add the second swap partition (`/dev/hdc2`) to the system.

Change `/etc/fstab` as follows:

```
/dev/hda2 none swap sw,pri=1 0 0
/dev/hdc2 none swap sw,pri=1 0 0
```

Create and activate the supplemental swap partition:

```
$ mkswap /dev/hdc2
$ swapon /dev/hdc2
```

Add a spare disk

As an optional step, you can add a spare disk to the *home* array:

```
$ mdadm /dev/md2 -a /dev/sdd1
```

Configure the boot loader

Configuring `lilo` to boot correctly from a RAID array can be a little tricky; for a situation similar to the one discussed here, a robust configuration is (only the boot related options are reported):

```
# The boot device (our boot partition)
#
boot = /dev/md0

# Devices which should receive a copy of the boot record
#
raid-extra-boot = "/dev/hda,/dev/hdc"

# The root device
#
root = /dev/md1
```

Before letting `lilo` overwrite the MBR's of your disks, you *should definitely use* the test mode:

```
$ /sbin/lilo -t -v
```

If everything is fine with the output, issue the usual:

```
$ /sbin/lilo -v
```

And reboot your machine.

RAID monitoring

md devices monitoring can be performed using `mdadm`; it can be easily configured to perform specific action on any state change of the md devices.

To receive an email alert for any of ours array, for example, the following command must be executed at boot time (by an init script or whatever suitable):

```
$ /sbin/mdadm -F -m root /dev/md0 /dev/md1 /dev/md2
```

The previous command notifies any disk or spare failure for any of the arrays specified as arguments. Please look at the `mdadm` man page for futher details and customizations.

References

[1] Michael Robinton: *Boot+Root+Raid+Lilo: Software Raid mini-HOWTO*

[2] Neil Brown: `mdadm` documentation

[3] John Coffman: *Notes for use of LILO on RAID installation*