HOWTO: Bootable Linux Software RAID-1 Array

Written by Chet McNeill. <chet at somedec dot com>

Version: June 24, 2005

Contents

1 Requirements

- Linux kernel >= 2.6.9
- mdadm
- grub
- Software RAID automount degraded mode kernel patch.
  This may be obtained from:
  `http://somedec.com/downloads/md-boot-degraded-2.6.12.1.diff`

2 Introduction

System administrators want the most reliable system that can possibly be had. In many cases hardware RAID controllers are either too expensive or simply unavailablef or a particular system.

There are many HOWTOs available on the Internet that describe several different schemes for utilizing Linux software RAID to provide mirroring of boot, root, and even other partitions. However, none of these have ever been robust enough to provide uninterrupted service under a great variety of circumstances. Most also were created using the 2.4.x series of Linux kernels, not taking advantage of new features available since the release of 2.6. For instance, most (if not all) proposed solutions involve mirroring partitions and treat a single mirrored partition as a usable standard-mode partition during the boot process.

Linux kernels in the 2.6.x generation have a new feature allowing the use of entire devices as components of a partitionable RAID-1 array. If one uses this new feature, then all data on the drive is mirrored at all times.

So the goal of this HOWTO is to put the partition table, and boot, root, and swap partitions on bootable RAID-1 mirror device. RAID device(s) should be assembled by the kernel before any filesystem is mounted.

3 Assumptions

- We will be using Gentoo installation as an example.
- We will assume two IDE drives, /dev/hda and /dev/hdb.
- For illustrative purposes we will also create other partitions using LVM2 on the RAID-1 array, providing a fully mirrored system.

4 Creating RAID device

1. Boot from Linux installation CD.
2. Load appropriate module

```
# modprobe dm_mod raid1
```

3. Create RAID-1 array of two drives:

   ```
   # mdadm -C -ap<#> -l1 /dev/md_d0 /dev/hda /dev/hdb
   ```

   Replace the <#> with the required number of partitions. Omit the number if the standard 4 is being used.

   I suggest using the standard four partitions.

4.1 Partitioning and creating filesystems

1. Partition the new RAID-1 array.

   ```
   # fdisk /dev/md_d0
   ```

   I suggest using something like the following, particularly if you're only using two drives.

   (a) Partition 1: /boot (128MB, type 83)

   (b) Partition 2: / (1GB, type 83)

   (c) Partition 3: swap (2 times physical memory size, type 82)

   (d) Partition 4: LVM (Remainder of array, type 8e)

2. Create file systems

   ```
   # mke2fs /dev/md_d0p1
   # mkreiserfs /dev/md_d0p2
   # mkswap /dev/md_d0p3
   # swapon /dev/md_d0p3
   # pvcreate /dev/md_d0p4
   # vgcreate vg /dev/md_d0p4
   # lvcreate -L 8G -n usr vg
   # lvcreate ...
   ```

3. Mount and use filesystems as normal.

   ```
   # mount /dev/md_d0p2 /mnt/gentoo
   # mkdir /mnt/gentoo/boot /mnt/gentoo/proc /mnt/gentoo/usr ...
   # mount /dev/md_d0p1 /mnt/gentoo/boot
   # mkswap /dev/md_d0p3; swapon /dev/md_d0p3
   # vgchange -ay vg; mount /dev/vg/usr /mnt/gentoo/usr; ...
   ```

4.2 Building your kernel

Of course, with Gentoo Linux you must build your own kernel. If you are not using Gentoo, you may be required to do the same. If you do, check the documentation for your distribution for details.

The kernel must meet these basic requirements:

- Make sure the kernel has RAID-1 (and LVM if using it) built in. You *cannot* use an initrd and modules. The kernel will only auto-mount raid arrays if the md module is built in. Both of these options are under Device Drivers->Multi-Device Support (RAID and LVM).
- The kernel must have the md-boot-degraded patch installed. This is definitely not part of any kernel through 2.6.12.1. The patch can be obtained from somedec.com (see above under Requirements for the URL).

To patch your kernel, obtain the patch file and follow these steps:

```
# cd /usr/src/linux
# patch -p1 < /path/to/patch/md-degraded-boot-2.6.12.1.diff
```

5 Installing packages

When installing packages, make sure that you install lvm2, mdadm (not raidtools), and grub.

Since you are using a 2.6 kernel, you will be using lvm version 2. So you need to make sure that the proper lvm tools package is installed.

Most users who are familiar with Linux software RAID are also familiar with raidtools. raidstart, raidstop, radhotadd, etc. are a time-honored tradition. However, the mdadm tool is a single that is *much* more powerful. Get used to it – you will love it.

In my testing, Lilo absolutely refused to be installed on a RAID-1 array. GRUB is the [only] way to go here.

6 Installing the boot loader

In testing I have found that it is important to reboot using the installation CD. You may not be required to do so, but proceed at your own risk. Be aware that grub will not complain and no errors will be reported. However, the resulting array simply will not boot. This condition is not fatal, and simply rebooting off of the install CD and re-installing grub will do the trick.

6.1 Rebooting your system

Be aware, that if you use LVM on a RAID-1 array you *must* do one of the following when booting from the installation CD:

- Upon reboot from the cdrom, add a kernel boot parameter "md=d0,hda,hdb" (replace the drives with values appropriate to your system); or
- After boot, make sure that dm_mod did not auto-load on boot. If an lsmod shows dm_mod, then unload it:

  ```
  # vgchange -an vg; rmmod dm_mirror dm_mod
  ```

The reason for the latter is that the init system on the install CD may scan for and find the LVM partition type and auto-load the LVM module, activating your volume group. Unfortunately, the LVM module just sees two drives with identical volume groups and simply ignores one of them. If LVM is not shut down, you will not be able to start your RAID array!

After booting (and possibly cleaning up), start your RAID array and volume group:

```
# modprobe raid1
# mdadm -A -ap /dev/md_d0 /dev/hda /dev/hdb
# modprobe dm_mod
# vgscan; vgchange -ay vg
```

Now you can remount all of the volumes and continue with your installation.

6.2 Installing grub

After chrooting into the /mnt/gentoo, install grub:

```
# grub
grub> root (hd0,0)
grub> install /boot/grub/stage1 (hd0) /boot/grub/stage2 p /boot/grub/menu.lst
grub> quit
# grub
grub> root (hd1,0)
grub> install /boot/grub/stage1 (hd1) /boot/grub/stage2 p /boot/grub/menu.lst
grub> quit
```

Notice that there are two distinct grub sessions. In testing I have found that you *must* install grub in two distinct steps.

Inside of your /boot/grub/menu.lst file, your boot configuration should look like this:

```
timeout 3
# By default, boot the first entry.
default 0
# For booting GNU/Linux title Gentoo
root (hd0,0)
kernel /bzImage-2.6.12 root=/dev/md_d0p2 ro md=d0,hda,hdb
```

Once that is done, you should be able to unmount all drives, reboot, and remove the installation CD. Congratulations!

7 Testing

Now you should test the system to make sure that all is working as expected.

7.1 Initial boot

After removing the installation CD and rebooting, grub should appear and happily boot your system. Of course, if you encounter a kernel panic, or system services start failing on boot you know there is a problem. Troubleshoot your initial install to get a working system.

Note that if grub reboots when loading itself, or if it freezes on phase 2, this is symptomatic of the installation of grub before rebooting (see section 6.2 above). Reboot from the installation CD, remount all devices, and re-install grub.

After booting up, scan the kernel output for status of your raid array(s):

```
# dmesg | grep md
```

You should see something like this:

```
md: Will configure md0 (super-block) from /dev/hda,/dev/hdb, below.
md: raid0 personality registered as nr 2
md: raid1 personality registered as nr 3
md: raid5 personality registered as nr 4
md: md driver 0.90.1 MAX_MD_DEVS=256, MD_SB_DISKS=27
md: Autodetecting RAID arrays.
md: autorun ...
md: ... autorun DONE.
md: Loading md_d0: /dev/hda,/dev/hdb
md: bind<hda>
md: bind<hdb>
md: kicking non-fresh hda from array!
md: unbind<hda>
md: export_rdev(hda)
raid1: raid set md_d0 active with 1 out of 2 mirrors
 md_d0: p1 p2 p3 p4
 md_d0: p1 p2 p3 p4
```

Also, check the current run-time status of the raid module:

```
# cat /proc/mdstat
```

You should receive output similar to the following:

```
Personalities : [raid0] [raid1] [raid5]
md_d0 : active raid1 hda[1] hdb[2]
       39082560 blocks [2/1] [UU]
unused devices: <none>
```

The key words to look for are 'active' and the '[UU]' section. Your array may report itself as 'reconstructing' if it has not fully synchronized yet.

If you see an underscore in place of one of the 'U's then one of your drives is not actively part of the array. See section 7.3.1 below on how to add the missing drive.

7.2 Software

The first and easiest way to test your array is to do it via software. After making sure that you have a working system, you should reboot. When grub's menu appears, hit 'e' to edit the command line. Move down to the kernel line and hit 'e' again to edit the kernel command line parameters. Change the text that reads 'md=d0,hda,hdb' to 'md=d0,hdz,hdb' (replacing your primary, or boot, drive with a non-existant drive designation). After hitting Enter to complete the editing session, press the 'b' key to continue booting.

At this point the raid module built into the kernel will try to assemble your RAID-1 array using a non-existant drive and your secondary, or mirror, drive.

If the kernel panics because it can not mount the root drive, then the cause is almost certainly that your kernel is missing the md-degraded-boot patch (see section 4.2 above).

If the system boots and everything looks normal then you are well on your way!

Now, if you check your kernel logs (dmesg | grep md) you should see something similar to:

```
md: Will configure md0 (super-block) from /dev/hdz,/dev/hdb, below.
md: raid0 personality registered as nr 2
md: raid1 personality registered as nr 3
md: raid5 personality registered as nr 4
md: md driver 0.90.1 MAX_MD_DEVS=256, MD_SB_DISKS=27
md: Autodetecting RAID arrays.
md: autorun ...
md: ... autorun DONE.
md: Skipping unknown device name: hdz
md: Loading md_d0: /dev/hdz,/dev/hdb
md: bind<hdb>
raid1: raid set md_d0 active with 1 out of 2 mirrors
md_d0: p1 p2 p3 p4
md_d0: p1 p2 p3 p4
```

Notice that it skips the unknown device. An unpatched kernel will notice the unknown device and abort at that point.

Of course, checking the run-time status of the RAID module will show the array running in degraded mode using only one hard drive:

```
Personalities : [raid0] [raid1] [raid5]
```

```
md_d0 : active raid1 hdb[1]
       39082560 blocks [2/1] [_U]
unused devices: <none>
```

If you wish to add the missing device to the array and start a resynchronization, issue the mdadm command:

```
# mdadm --manage --add /dev/md_d0 /dev/hda
```

If you are going to continue testing with the hardware section below, then you probably do *not* want to start the resync yet.

7.3 Hardware

Now comes the real test. Shut down your system after trying the software test above. Now uplug the primary drive. Now we will force the machine to load grub from the second hard drive, load the kernel, and reconstruct the degraded RAID-1 array.

Power up the machine. Both grub and the kernel should load without any noticable difference! In fact, the system should boot completely normally. The only differences will be a slight glitch in the kernel logs similar to the software test above. Checking the kernel logs for RAID references (dmesg | grep md) should return something like:

```
md: Will configure md0 (super-block) from /dev/hdz,/dev/hdb, below.
md: raid0 personality registered as nr 2
md: raid1 personality registered as nr 3
md: raid5 personality registered as nr 4
md: md driver 0.90.1 MAX_MD_DEVS=256, MD_SB_DISKS=27
md: Autodetecting RAID arrays.
md: autorun ...
md: ... autorun DONE.
md: Skipping unknown device name: hdz
md: Loading md_d0: /dev/hdz,/dev/hdb
md: bind<hdb>
raid1: raid set md_d0 active with 1 out of 2 mirrors
 md_d0: p1 p2 p3 p4
 md_d0: p1 p2 p3 p4
```

The run-time status will be identical to the software test above:

```
Personalities : [raid0] [raid1] [raid5]
md_d0 : active raid1 hdb[1]
       39082560 blocks [2/1] [_U]
unused devices: <none>
```

7.3.1 Restoring your RAID-1 array

After shutting down and re-connecting your primary hard drive, reboot your system. You might think that the RAID array will now begin resynchronization. You might, then, be surprised that it, in fact, does not.

Checking the run-time status of the RAID module shows:

```
Personalities : [raid0] [raid1] [raid5]
md_d0 : active raid1 hdb[1]
       39082560 blocks [2/1] [_U]
unused devices: <none>
```

You must add the missing device back into the array using the following command:

```
# mdadm --manage --add /dev/md_d0 /dev/hda
```

Voila! Your array is now synchronizing.

8 Counter-indications

The only drawbacks that are immediately obvious to this specific solution are:

- A kernel patch & build are required. While Gentoo users will likely not have any problem here, users of other distros may lack kernel building experience.

- Replacing a failed hard drive means that the RAID-1 array will suck up the entire drive. For example, if one were to mirror two 40GB drives, and replace a failed drive later with an 80GB drive, 40GB on the new drive is completely unusable. When using RAID-1 partitions, the partition table from the surviving drive can be duplicated on the replacement drive, and whatever space is remaining can still be partitioned and used. On the other hand, with this method *no partitioning of the replacement drive is required!* The partition information is part of the RAID array and will automatically be copied to the new drive during synchronization.