

Installing Debian with SATA based RAID

Later yet news(1/07)

The new Etch installer does all this for you and more -- I leave this up for people that want to transition from an existing system to raid without reinstalling - might be easier to back-up and reinstall. This page also provides clues as to how to recover from many situations - so I will leave it up.

Late breaking news (4/05)

The newer Debian installers have Raid (and also LVM) working, so this guide should not be needed unless you are transitioning from a standard set up to raid without reinstalling. It might also be helpful if you are debugging a broken raid.

Now for 2.6 kernel version

I've read that there will soon be an installer that will do raid installs and perhaps even support SATA, but today it is manual. My install on a Intel D865PERL mother board got 'interesting'. The last Debian (beta 4) testing installer does support SATA as does a version of Debian/Libranet, but going on to RAID is a manual task.

The basic idea was to Install all on one drive, then partition the second drive with just the same sizes. Install mdadm (the new and improved replacement for raidtools). I'm assuming we are using clean drives. If one re-uses previously used disks the superblocks must be zeroed (--zero-superblock option to mdadm) before adding the partitions to the array.

This guide was produced using a Tyan S2875 Moter board.

The whole mess is harder than it should be and I hope this page becomes the basis for someone's automating script (hint hint).

Overview of steps

- Install Debian on first drive (/dev/sda)
- Creating a degraded RAID1 array on disk 2 (/dev/sdb)
- Update initrd

- Copying the Debian installation over from disk 1 (sda to sdb)
- Fix fstab on /Dev/md2
- Add disk 1 to the degraded array to create final raid array
- Update initrd again
- Produce /etc/mdadm/mdadm.conf
- Setup the monitoring daemon

In this example Disk1 and disk2 are /dev/sda and /dev/sdb respectively.

The partitions I used are:

```
/boot 200m
/swap 1Gig raid0
/ rest of drive
```

A note on raid systems and swap:

It is possible for a system to crash if the swap area drive fails. If this is not a concern you could also set it up as simple swap areas or as raid0 to improve performance. (Raid1 also provides faster reads as a side benefit, but there is the cost slower writes - but swap tends to be read more times than written so you might still come out ahead.)).

Install Debian on first drive

I used the following partitions

Device	Size	Id	Eventual mount point
/dev/sda1	200M	83	/boot/
/dev/sda5	1G	82	swap
/dev/sda6	100G	83	/

I'm assuming you know how to do the above by running fdisk from the install of your choice.

After installing:

```
#apt-get install wajig (you don't want to be without wajig - it makes apt and
dPKG user friendly!! )
#wajig install mdadm
#wajig install rsync
#wajig install e3
```

```
#wajig install jfsutils
Update kernel to kernel-image-2.6.8-1-k7-smp
```

Partition Drive2 with fdisk

Device	Size	Id	Eventual mount point
/dev/sdb1	200M	fd	/boot/
/dev/sdb5	1G	fd	swap
/dev/sdb6	100G	fd	/

Create the raid devices

```
# mdadm --create /dev/md0 --level 1 --raid-devices=2 missing /dev/sdb1
```

The above line is the long version of the next line just for reference. .

```
#mdadm -Cv /dev/md0 -l1 -n2 missing /dev/sdb1
#mdadm -Cv /dev/md1 -l1 -n2 missing /dev/sdb5
#mdadm -Cv /dev/md2 -l1 -n2 missing /dev/sdb6
```

This creates 3 degraded RAID1 devices (for /boot, swap, and /) consisting of a dummy drive "missing" (why was this so hard to figure out!)

Now, a cat of /proc/mdstat will show your degraded raid devices are up and running

```
#cat /proc/mdstat
Personalities : [raid1]
md1 : active raid1 sdb5[1]
      979840 blocks [2/1] [_U]

md0 : active raid1 sdb1[1]
      192640 blocks [2/1] [_U]

md2 : active raid1 sdb6[1]
      159661888 blocks [2/1] [_U]

unused devices: <none>
```

Note how one drive in all three cases is missing ("_") as opposed to 'Up and running' ("U").

Creating the file systems and Mount them

```
#mkfs.jfs /dev/md0
#mkfs.jfs /dev/md2
#mkswap /dev/md1
#mkdir /mntroot
#mkdir/mntboot
```

```
#mount /dev/md2 /mntroot
#mount /dev/md0 /mntboot
```

Fix up initrd

If you are using a SATA drive you pay attention!

edit /etc/mkinitrd/mkinitrd.conf and change:

```
MODULES=most
```

to

```
###MODULES=most
MODULES=dep
```

and

```
ROOT=probe
```

to

```
###ROOT=probe
ROOT="/dev/md2 jfs"
```

This tells init to use what it takes to boot off of a raid device not the /dev/sda device currently used.

Now run

```
#mkinitrd -o /boot/initrd.img-2.6.8-1-k7-smp-md 2.6.8-1-k7-smp
```

Edit /boot/grub/menu.lst

Copy the old listing below and paste it below:

```
title Debian GNU/Linux, kernel 2.6.8-1-k7-smp
root (hd0,0)
kernel /vmlinuz-2.6.8 root=/dev/sda6 ro
initrd /initrd.img-2.6.8
savedefault
boot
```

Now edit the new stanzas noting the changes in bold:

```
title Debian GNU/Linux, kernel 2.6.8-1-k7-smp-md Disk1
root (hd0,0)
kernel /vmlinuz-2.6.8 root=/dev/md2 ro
initrd /initrd.img-2.6.8-1-k7-smp-md
savedefault
```

boot

Now make a copy of the New stanza you just made and make the changes in bold:

```
title Debian GNU/Linux, kernel 2.6.8-1-k7-smp-md disk2
root (hd1,0)
kernel /vmlinuz-2.6.8 root=/dev/md2 ro
initrd /initrd.img-2.6.8-1-k7-smp-md
savedefault
boot
```

Producing /etc/mdadm/mdadm.conf

edit /etc/mdadm/mdadm.conf

The top line in this example should be

```
DEVICE /dev/sda* /dev/sdb*
```

Then run:

```
mdadm --detail --scan >> /etc/mdadm/mdadm.conf
```

This should produce a proper file

Copy over everything to the degraded raid

```
#rsync -auHxv --exclude=/proc/* --exclude=/sys/* --exclude=/boot/* --
exclude=/mntboot --exclude=/mntroot/ /* /mntroot/
#mkdir /mntroot/proc /mntroot/boot /mntroot/sys
#chmod 555 /mntroot/proc
#rsync -auHx /boot/ /mntboot/
```

Make Changes in /mntroot/etc/fstab

Changes in bold

```
# /etc/fstab: static file system information.
#
# <file system> <mount point> <type> <options> <dump> <pass>
proc /proc proc defaults 0 0
/dev/md1 none swap sw 0 0
/dev/md0 /boot jfs defaults 0 2
/dev/md2 / jfs defaults,errors=remount-ro 0 1
/dev/hda /media/cdrom iso9660 ro,user,noauto 0 0
/dev/fd0 /media/floppy auto rw,user,noauto 0 0
/dev/hda /cdrom iso9660 ro,user,noauto 0 0
```

Reboot

If it won't work you should be able to boot off of the old grub listing

At this point the box should be running off the degraded RAID1 devices.on the second drive (/dev/sdb)

Check that everything works

If all is OK you should chnge the partition types of the first drive (/dev/sda) - but you will lose all data!!

Device	Size	Id	Eventual mount point
/dev/sda1	200M	fd	/boot/
/dev/sda5	1G	fd	swap
/dev/sda6	100G	fd	/

Attach the original drive's partitions to the existing (degraded) RAID arrays:

```
# mdadm /dev/md0 -a /dev/sda1
# mdadm /dev/md1 -a /dev/sda5
# mdadm /dev/md2 -a /dev/sda6
#cat /proc/mdstat
Personalities : [raid1]
md0 : active raid1 sda1[0] sdb1[1]
      192640 blocks [2/2] [UU]

md1 : active raid1 sda5[0] sdb5[1]
      979840 blocks [2/2] [UU]

md2 : active raid1 sda6[2] sdb6[1]
      159661888 blocks [2/1] [_U]
      [===>.....] recovery = 17.9% (28697920/159661888)
finish=56.4min speed=38656K/sec
unused devices: <none>
```

After a while all devices are in sync:

```
# cat /proc/mdstat
Personalities : [raid1]
md0 : active raid1 sda1[0] sdb1[1]
      192640 blocks [2/2] [UU]
md1 : active raid1 sda5[0] sdb5[1]
      979840 blocks [2/2] [UU]
md2 : active raid1 sda6[2] sdb6[1]
      159661888 blocks [2/2] [UU]
unused devices: <none>
```

DO Not Reboot!!!

Regenerate initrd

Yes, this needs to happen again after the raid is complete!

```
#mkinitrd -o /boot/initrd.img-2.6.8-1-k7-smp-md 2.6.8-1-k7-smp
```

This is still a needed step! If initrd.img is generated with a degraded array you will end up with a degraded array every time you reboot! This includes when you install a new kernel-image pkg which runs a script that produces a initrd.img! You have been warned!

Regenerate /etc/mdadm/mdadm.conf

```
edit /etc/mdadm/mdadm.conf
```

Delete all but the top line

```
DEVICE /dev/sda* /dev/sdb*
```

Then run:

```
mdadm --detail --scan >> /etc/mdadm/mdadm.conf
```

Your final file should look like:

```
DEVICE /dev/sda* /dev/sdb*
ARRAY /dev/md0 level=raid1 num-devices=2
UUID=4d58ade4:dd80faa9:19f447f8:23d355e3
    devices=/dev/sda1,/dev/sdb1
ARRAY /dev/md1 level=raid1 num-devices=2
UUID=3f1bdce2:c55460b0:9262fd47:3c94b6ab
    devices=/dev/sda5,/dev/sdb5
ARRAY /dev/md2 level=raid1 num-devices=2
UUID=7dfd7fcb:d65245d6:f9da98db:f670d7b6
    devices=/dev/sdb6,/dev/sda6
```

Note - you need the top DEVICE line and the lower devices lines

Configure grub to boot from both drives

Install the MBR on the second disk

```
#grub --device-map=/boot/grub/device.map
>> root (hd0,0)
>> setup (hd0)
>> root (hd1,0)
>> setup (hd1)
>> quit
```

Have Grub point to your MD root device. Note that things after the '#' symbol make a difference here (note to grub developers: this confuses people why not use a different delimiter?)

```
Edit /boot/grub/menu.lst
```

```
## default kernel options for automagic boot options
## If you want special options for specifiv kernels use kopt_x_y_z
## where x.y.z is kernel version. Minor versions can be omitted.
## e.g. kopt=root=/dev/hda1 ro
# kopt=root=/dev/sda6 ro
```

TO

```
## default kernel options for automagic boot options
## If you want special options for specifiv kernels use kopt_x_y_z
## where x.y.z is kernel version. Minor versions can be omitted.
## e.g. kopt=root=/dev/hda1 ro
# kopt=root=/dev/md2 ro
```

Setup the monitoring daemon

Just run:

```
dpkg-reconfigure mdadm
```

Test Test and Test

Test boot from both drives

Kill a drive and see if you get a email about the event.

Write up a step by step procedure to restore from a drive outage. (send a copy this way for this page!)

You should be all finished!

Please send notes of any typos/corrections to the email address below.

Special thanks to Onni Koskinen of Finland, whose gentle yet expert emails removed several glaring errors on this page and resulted in a vastly improved document.

Growing Raid1 Arrays

One can replace drives in a Raid1 array with larger ones (with larger partitions) and end up with an array that is smaller than the partition. There is a poorly documented 'grow' command (mdadm -G) that allows one to increase the size. I could not tell from the documents if the --size=?? option is needed (which should be in blocks) - I don't see how one is supposed to know what the new number should be.

If the Grow command works there is a second problem - ext partitions will stay the same size - jfs partitions can be extended with the mount command:

```
mount -o remount,resize /mount/point
```

I have not tested this - if someone else has - please email me what you have learned.

Re-adding Faulted drive

First, look at proc:

```
cat /proc/mdstat
Personalities : [raid1]
md1 : active raid1 sda2[2](F) sdb2[1]
      70645760 blocks [2/1] [_U]

md0 : active raid1 sda1[0] sdb1[1]
      9767424 blocks [2/2] [UU]
```

```
unused devices: <none>
```

This shows raid md1 has drive sda2 stopped with a fault.

To re- add:

```
# mdadm /dev/md1 -r /dev/sda2
mdadm: hot removed /dev/sda2
# mdadm /dev/md1 -a /dev/sda2
mdadm: re-added /dev/sda2
```

Now you will see it regenerate in mdstat:

```
# cat /proc/mdstat
Personalities : [raid1]
md1 : active raid1 sda2[2] sdb2[1]
      70645760 blocks [2/1] [_U]
[>.....] recovery = 0.3% (268800/70645760) finish=21.8min
speed=53760K/sec
```

```
md0 : active raid1 sda1[0] sdb1[1]
      9767424 blocks [2/2] [UU]
```

```
unused devices: <none>
```

If you have to re-add a drive more than once you need to find out why.

Q & As

Re: section on configuring GRUB: will GRUB automatically boot from the good drive in the event of a disk failure?

Yes, IF you install Grub on both drives and your BIOS will roll over to the first bootable drive.

How do you see what is in a initrd file?

```
mount -o loop /tmp/myinitrd /mnt/myinitrd
```

Notes from others:

I would like to add that this in fact maps the device names /dev/mdx to the physical drives. In your setup, this does not matter because you have md0, md1, md2, ... However when you produce a setup of for example:

```
/dev/md0 for /  
/dev/md4 for /var  
/dev/md5 for /mnt
```

And you do not create the mdadm.conf file, your device names will differ into:

```
/dev/md0 for /  
/dev/md1 for /var  
/dev/md2 for /mnt
```

More info can be found in the file /etc/init.d/mdadm-raid that is run at bootup before the mounting starts.

In their, you will run:
mdadm -A -s -a.

However if the /etc/mdadm/mdadm.conf file is empty, it will run mdrun which will assign numbers md0, 1, 2, and so on.

A second note when using udev on 2.6 kernels (as far as I know, not tested on 2.4), you need to point you DEVICE line in /etc/mdadm/mdadm.conf to the "static" entries as otherwise the system will not find it at boot time.

So for example in case of UDEV:
your /etc/mdadm/mdadm.conf should look like:

```
DEVICE /dev/.static/dev/sda* /dev/.static/dev/sdb*  
ARRAY /dev/md0 level=raid1 num-devices=2 spares=1  
UUID=5c88a427:c5655de8:b97d2fa2:f5b1627e  
    devices=/dev/.static/dev/sda1,/dev/.static/dev/sdb1
```