INTERLAB
Tel:+331 3971 7337 – Fax: +331 3928 9008

RAIDTOOLS : Stripe 0 HOW TO

http://www.tldp.org/HOWTO/Software-RAID-HOWTO.html – Linux Software RAID HOWTO

http://oss.sgi.com/projects/xfs/ – SGI XFS File system website

http://www.bitmover.com/lmbench/ – LMbench Linux tools for performance analysis.

1. Plugging in, and turning on!

$> cat /proc/scsi/scsi

To check that all ressources came up right

And this should report something like the following:

Attached devices:

Host: scsi0 Channel: 00 Id: 05 Lun: 00
Vendor: ATA Model: VideoRaid RTR Rev: 1.58
Type: Direct-Access ANSI SCSI revision: 03
Host: scsi1 Channel: 00 Id: 06 Lun: 00
Vendor: ATA Model: VideoRaid RTR Rev: 1.58
Type: Direct-Access ANSI SCSI revision: 03

2. Linux software level RAID

Now we have a system with the Medea array attached. The next step is to take the two halves of the array and stripe them together into one large array. Since the each section of the array already has it's own RAID level 5 protection, we don't actually need any redundantcy when we stripe the two halves together. So we will be using Linux software RAID 0 (which is a sort of misnomer as RAID 0 simply stripes disks together without any redundantcy) to stripe the two halves together in what is known as a RAID 5+0 configuration. This text, however, is not meant to be a lesson in Linux software RAID, so

please check out the Linux software RAID HOWTO sited in the "Suggested Readings" section of this document.

First you will need to edit the **/etc/raidtab** file to set up the raid settings for the software array. It should look similar to the following:

```
raiddev /dev/md0
raid-level 0
nr-raid-disks 2
persistent-superblock 1
chunk-size 64

device /dev/sda
raid-disk 0
device /dev/sdb
raid-disk 1
```

Note: sda and sdb are the two halves of the array. These may be different on your system, so please check your system specifications to assure they are correct.

3. Raid init

Once you have edited the raidtab file, you will need to initalize the RAID device. This is done via the following command:

```
$> mkraid /dev/md0
```

This will run for a few moments and should return without error. Note: Depending on your Linux distrobution, you may need to add a "raidstart /dev/md0" to your startup scripts.

4. Filesystems

When working with high performance video on a Linux box, the filesystem of choice is SGI XFS. It is extremely fast and implements several advanced features need to atain the bandwidth for high end video. Unfortunatly, XFS does not yet come stock in the vanilla Linux kernel, so some amount of kernel patching is nessicary. You will need to download the patches an tools from SGI (see "Suggested Readings" section) in order to continue. This text assumes that you have already downloaded and installed the tools as well as have already patched the kernel (some distrobutions already have prepatched kernels).

Once XFS is ready to go on the box, formatting your new array is as easy as typing the following commands:

```
$> mkfs.xfs /dev/md0
```

You will also need specify the mount point by adding a line to your **/etc/fstab** file with something like the following (as well as creating the directory for the mount point):

```
/dev/md0 /video xfs defaults 0 0
```

And a final "mount /video" will bring your newly created filesystem online.

3. Performance Tweaking

The following are a few tips I have found to get the most bandwidth performance out of your array. Performance enhancements of upto 60% are possible, so please read on!

3.2 Adjusting the RAID settings

The next variable in the performance tweaking puzzle is the software raid chunk size. This is the size of the raid stripe or the amount of data that is written to the disk before it goes onto the next disk. Adjusting this settings will allow you to take advantage of buffering in the array or drives. By setting this to an optimal level the disk can quickly ingest the data and then write the data while the software raid algorithm has moved on to the next disk.. this essentially doubles the datarate (sometimes)!

So take a look at the /etc/raidtab again. Change the change the line that says:

chunk-size 64

to

chunk-size 512

Now you will need to stop the RAID array and re-make the raid device with the following commands:

$> raidstop /dev/md0

$> mkraid --force /dev/md0

3.3 Adjusting the file system

Since we have changed the transfer size to 2k bytes in the section above, we can tell XFS to use this new size when we format the filesystem. This is does with the following command:

$> **mkfs.xfs -d sectsize=4096 -l sectsize=4096 /dev/md0**

This creates an XFS filesystem with a sector size of 4k, instead of the standard 512 bytes.

4. Performance Testing

Performance testing is a very important step in assuring your array will meet your datarate needs. It is important to not only test read speeds, but write speeds as well. These two number can vary widely based on the performance tweaking that you do. The method here is: **change one thing, test, repeat**!

4.1 To SD or to HD?

It is very important to understand exactly what you want to get out of a disk array. Will you be working with mostly SD video, mostly HD video, or both? Certain adjustments to the setup will, for instance, benefit HD but not SD. (some may even slow down one and speed up the other). To get an idea of what sort of datarates you should be looking for, consult my handy uncompressed video data rate chart at:

4.2 Tools for the Job

All the testing in this document is based around lmdd a child of the LMbench Linux benchmarking suite (see "Suggested Readings" section). It is also worth noting that my company (SpectSoft) has a freely available (GPLed) disk benchmarking tool for testing disk performance across all the standard video frame sizes and rasters.

In any case, go ahead and download the LMbench software, comple, and lets get