

How I got to software RAID 10

(Or my trip down the yellow SATA cable road)

A high performance, highly reliable software RAID solution for Dotsch/UX, Ubuntu, Xubuntu. Arguably one of the best RAID configurations available and likely the very best when using software RAID.

After I started converting my dedicated, headless number crunchers (BOINC) over to be diskless clients under a Dotsch/UX server I started to worry about them all going down for an extended period if the one drive they all got their OS image from went belly up. This started me on a multi-month trek of looking at and trying a couple different HDD configurations. Below is the story of my trip 'down the yellow sata cable'. At various points I proved to myself that I could be either the Tin Man, the Lion or the Scarecrow. Included later are various tidbits of information I found along the way with links to a couple of "How To" articles that I used.

I initially set up and installed Dotsch/UX on a single IDE 120GB drive. I created partitions for swap, /, /home and /diskless on it. I then spent all my time with the DHCP server setup and getting an understanding of how to implement my largely fixed IP network here and eventually turned off the DHCP server in my router. Being a relative newbie to Linux and having never set up DHCP outside of a router I made this process way harder than it probably needed to be but I got a good education out of it. I also tested Lar's patience more than once with my lack of a brain and with our occasional language gaps he was too polite to tell me "Look idiot, it'll do that itself you don't need to change everything."

Once that was done and had been handling my LAN machines for about a week I started to add diskless clients. I already had a number of headless dedicated crunchers running BOINC under Xubuntu so it was really just a matter of setting a couple of them to no-new-work and letting them finish up their BOINC work units. I then just rebooted to BIOS, set them to LAN BOOT / PXE, disabled the on-board disk controllers and unplugged the drive. I then ran the diskless client add tool in Dotsch/UX, updated my DHCP conf files to comment out the old entry for that machine and let it boot up as a diskless client, reattached to BAM and it was off and running.

After converting a couple (2 or 3) of them to be diskless clients I noticed that the HDD light on the Dotsch/UX server seemed to be ON a lot. As it turned out from later actual stats gathering I was being alarmist but this got me thinking about the fact that I was putting all my eggs in one basket so to speak and that I had NO backup solution in place. Before this if a HDD went out it took down 1 cruncher and I used to keep 2 or 3 extra old HDDs on the spare

parts shelf. I'd set up my "Xubuntu/BOINC/x11vnc/apt-cacher client" install so many times I could do nearly all of it from memory so this didn't seem like a big deal to me. However, now if that drive in the Dotsch/UX server bellied up I was taking it and the disk-less clients to ground zero all at once. This bothered me. After all, one must maintain ones RAC, right?

Interim step 1 – Separate Drives

Well the first step I took was to look thru the spare parts shelf and see what we had. It seems I had a surplus of 40GB SATA drives. I had ebay'd myself some small lots of these for building 'basket crunchers' when it started to become hard to find old 10 / 20 GB IDE drives to use in them.

So 'mostly wasted time step one' was to change the Dotsch/UX server over from the single 120GB drive to use two 40GB SATA drives. Since /diskless was a separate partition (and I HIGHLY recommend this as a general installation pattern for setting up a Dotsch/UX server) I just added one of the 40GB drives to the machine, plugged in a CD drive and booted from the live CD. There I used either Acronis True Image or DD (I don't remember which but I've used both) to clone /diskless to the 'new' 40GB drive. I then used the partition editor (gparted) to expand that to be a single full disk partition and adjusted /etc/fstab to mount that drive as /diskless. There are a thousand threads and pages on cloning partitions. You could start [HERE](#).

With that done, tested and running for a day or two, I installed a 2nd 40GB drive in the Dotsch/UX server and now cloned the OS partitions over to the 2nd 40GB drive. Well I'm feeling a wee bit better about this now but I still need to get a backup solution implemented.

I got back on my merry path of running some more crunchers dry and converting them over to be disk-less clients.

Interim step 2 – RAID 1, 'mirror'

More clients added and now I'm worrying about that single /diskless partition drive. No, I still don't have a scheduled backup system running either. The HDD light seems to almost stay on now and remote access to them via the BOINC GUI is becoming sluggish. I wish I'd kept better notes but this must have been at around the 4~6 disk-less client stage.

I'm also playing with BackupPC on my SAMBA server but so far no luck getting it to work.

This is the point in time that I started looking at hardware RAID and that led me to looking at software RAID. *See General RAID Stuff section further down.*

I had a twin to the 40GB SATA150 drive that was mounted at /diskless so I physically mounted it and proceeded to make the pair into a RAID 1 mirror for /diskless. At this point my drives were:

- 1) /dev/sda (OS drive)
- 2) /dev/sdb (current /diskless)
- 3) /dev/sdc (future /diskless mirror drive).

Remember that at this point in time my /diskless was a single partition / drive so I did not need to use a live CD. I simply shutdown all the diskless clients, commented out the mount of /diskless in /etc/fstab and rebooted the Dotsch/UX server. It'll complain about not finding the tftboot and /diskless/\$hostname files but it'll boot up and run fine.

The rough steps covered in the 'How To' are:

- 1) Copy or Tar the /diskless partition off to some other partition/drive. (make **sure** you preserve permissions with -p switch on either).
- 2) Set up the HDDs that are to be in the array with fdisk. (I don't believe this is required*)
- 3) Establish array with mdadm -c (--create).
- 4) Reboot (array should start to sync).
- 5) Format the array /dev/md0 (believe it or not you CAN format it while it's rebuilding).
- 6) Edit /etc/mdadm/mdadm/conf ("sudo mdadm --detail --scan >> /etc/mdadm/mdadm.conf").
- 7) Create a new entry in /etc/fstab to mount /dev/md0 at /diskless.
- 8) Copy the original /diskless data from step 1 over to your new /dev/md0 RAID 1 array.
- 9) Use "watch -n 5 cat /proc/mdstat" to see the sync running. Once done "mount -a" or probably better to reboot, check that that all is as it should be and re-start the clients.

** If the drive is to be a single partition. Read the thread at the link in the RAID10 section to see the "fdisk -l" differences of formatting the partitions here vs just formatting the array after it's defined. I suspect formatting the partition first as described in the PersonalBytes write up may make some mdadm.conf entries optional. I ended up redoing my arrays later to figure this out and the last time thru I omitted this step and just formatted the array. I have all the devices defined in mdadm.conf.*

There are a slew of How-To articles for this. The one I used may not be the best and it's all

command line but it's also a very simple scenario of non-boot array like I needed for /diskless. Find it at [PersonalBytes](#). In this one he doesn't cover setting up /etc/mdadm/mdadm/conf but it's really just the single command above in step 7.

There are several out there that use an Alternate-CD and GUI to do the whole process. These may very well be less trying. The main reason I used the above is he was not doing a multi-partition OS drive as the following seem to be doing.

[Kuparinen](#)

[Ubuntu_doc](#)

[Linuxconfig](#)

Final configuration step 3 – RAID 10, the 'stripe of mirrors'

With a slight improvement in read performance from my RAID 1 setup but mostly being less worried about the /diskless drive just going out on me I was back on my merry path of running some more of the existing crunchers dry and converting them over to be disk-less clients. I ran with that configuration and finished off the conversion of the headless crunchers to be disk-less clients. Before I'd finished converting them all over the remote GUI response in the BOINC manager had become painfully slow. For example, click on a project in the first tab, wait for it to highlight, click on the Update button, wait for it to show “schedule request pending”...

There was another issue now also. With 12 of my 'heavy' clients converted I was now running over 95% capacity of the mirror. This is way higher than I am used to running any drive at. I did several time consuming and often only semi-productive and some times destructive work of removing packages (via Synaptic) on each client. I also relocated /diskless/master.x64 to the /opt/Dotsch_UX directory for a while. I also modified my logrotate setup. Eventually I got the usage back down to around 90~92%. I still wasn't real happy with that. I would've been much happier with a usage of under 80%.

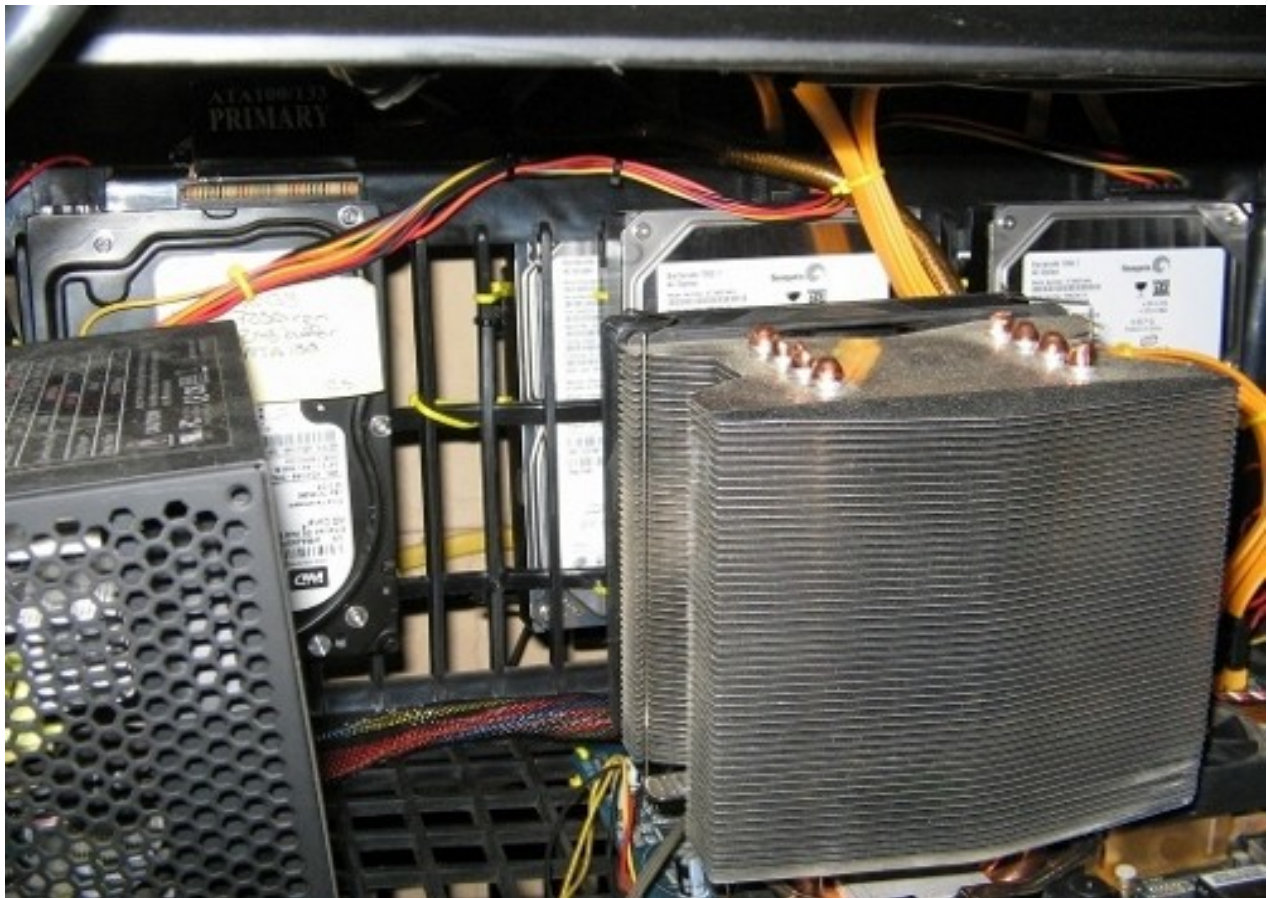
I started to think about having two Dotsch/UX servers and splitting the load that way but that would mean some DHCP challenges and probably sub-netting. Not something I really felt like taking on at the time. Lar's helped me to set up and look at some iostat log output to see where the load was. A couple things showed up. The first being that the disk-less clients themselves didn't seem to be suffering as much as I was. Secondly, it appeared to be more of a write bottleneck than a read problem. Lar's had mentioned RAID 5 and RAID 10 to me and off I went back into RAID research mode.

I now had a nice pile of extra HDDs laying around and in there found two more that were exact model matches and fairly close date matches to the two in my RAID 1 mirror!

This led me to conclude that my newly found 4 matching drives were a sign from above that my answer was RAID 10. It did seem that it would address both of my problems as it would share the write load across two mirrors (functionally it shares reads across all 4 drives) and would double my available storage space. All this for FREE! It doesn't get much better.

Well it turns out getting 5 HDDs mounted where cables can reach them, onto a Container Store plastic vegetable basket (my cruncher 'cases') was going to be a bit of a physical challenge. Certainly it's one that could be overcome with some time on the work bench and a fresh bag of 1000 zip ties. Oh then there was the fact that the mobo I was using only had 4 SATA ports.

After a couple of hours I managed to get 5 drives firmly tied down, cables routed / bundled and still have a bit of little bit of airflow on the backside of my basket.



See the the GigaByte “yellow SATA cables” in C34, the Dotsch/UX server. ;-) The OS drive is the black WD400, IDE drive on the left. The 4 Seagate SATA150 drives are the two inside the basket to the right, behind the HSF and the two hanging on the outside of it. That's a 80+

bronze certified Rosewell ~430w PSU in the front left. Gigabyte G31 chipset based uATX mobo under the big HSF.

Note: Some of you probably have nice air-conditioned server rooms. My crunchers live in a non-A/C'd space, avoided by dogs and people throughout the long Texas summer, affectionately known as the "server oven". Pics at skipsjunk.net

The rough steps I needed to do were:

- 1) Clone the existing SATA OS drive over to an IDE drive to free up a SATA port (live CD needed for this step only, adjust /etc/fstab to boot from the cloned over drive).
- 2) Create a 2nd mirror with the the two 'new' drives with mdadm -c (/dev/md1, this can be done w/o being in the live CD. For me this was /dev/sdd and /dev/sde).
- 3) Format the new mirror (/dev/md1) to ext3.
- 4) I brought all the clients down at this point and "failed" one drive in the existing array (/dev/md0) with "mdadm /dev/md0 --fail /dev/sdb --remove /dev/sdb". *Here I made the paranoid mistake of actually unplugging that drive and this caused the other drives to change device assignments. Leave it plugged in to avoid headaches.*
- 5) Create a stripe across the two mirrors (RAID 0 array, /dev/md2) with mdadm -c consisting of /dev/md0 and /dev/md1.
- 6) Format the new /dev/md2 array (your data is still safe on the failed out /dev/sdb).
- 7) Make two temp mounting points. Mount /dev/md2 to /mount1 and the drive that was taken out of the original mirror to /mount2.
- 8) Copy the original /diskless data (from /dev/sdb failed in step 4) into the new RAID10 array ("rsync -avHxl --progress --inplace --exclude 'lost+found' mount2/ mount1/").
- 9) Unmount /dev/sdb and /dev/md2.
- 10) Add /dev/sdb back to /dev/md0 ("sudo mdadm /dev/md0 --re-add /dev/sdb").
- 11) Edit /etc/mdadm/mdadm.conf as needed to include info on all 3 arrays ("sudo mdadm --detail --scan >> /etc/mdadm/mdadm.conf"). Examples at the end.
- 12) Replace the mount to /diskless of /dev/md0 with /dev/md2 in /etc/fstab.
- 13) Reboot to check that that all is as it should be. Use "watch -n 5 cat /proc/mdstat" to see the sync running. Once the sync/rebuild is done re-start the clients.

My original source of information and the details on converting a RAID 1 mirror to a RAID 10 stripe of mirrors can be found in a GREAT post here at LinuxQuestions.com.

RAID 10 Notes:

- 1) A caution I have to add is - double check your device assignments. On my mobo if I unplug what was, let's say, /dev/sda then the drive that was /dev/sdb becomes /dev/sda and they all move up a notch. You can correct for this if it occurs by editing the available devices and array members in /etc/mdadm/conf but be aware of this and **if at all possible do all array building with the drives plugged in where the will be when it's finished and running.**
- 2) I've had NO luck getting UUIDs to stick for use in /etc/fstab and have reverted to specifying the dev/md2 to be mounted. I may have had this working with RAID 1 but not with RAID 10.
- 3) Example mdadm.conf file and outputs are listed at the end.
- 4) Yes, I finally got scheduled backups going. After I got the RAID 10 up I finally installed sbackup to get something going but continued to play with BackupPC for a bit longer till I finally gave up on it. Simple Backup (sbackup) seems to be working fine and will ssh the tar'd backup over to the RAID 1 mirror on my SAMBA server (an Xubuntu basket cruncher on the same LAN) at /mnt/backups/\$hostname.

Dotsch/UX configurations based on # of 'heavy' clients -

First off this is limited to just sizes and configurations I have touched. I can imagine others using a couple of larger modern drives with the OS and /diskless all on the same drives. I've just got no experience with that. Also realize that the load a disk-less client puts on the server can vary quite a bit.

My experience is all based on the disk-less clients being Intel C2D quads with 2 x 1 GB of memory in each. Each quad is OC'd to between 3.02 and 3.57Ghz. Each one is running the same set of around 16 projects via BAM. These projects include some that can use a fair amount of disk space. The #1 space hog has to be CPDN. Of the projects I run CPDN is the run away leader in space usage with it using about 7 or 8 times what the #2 project, Einstein uses. #3 & #4 is about a tie between Milkyway and WCG both using about 15% less than Einstein.

So I would think these headless crunchers qualify as a 'heavy load' client both in terms of disk/lan activity and space usage.

If you took the same hardware and only ran, let's say 2 projects, what would change? I would guess the disk space usage would drop off a bit just due the reduction in the number of executables needed but I suspect that's about all that would change.

Dual Core machines? The simplistic theory would be that they would put about 50% of the disk and LAN activity load on the server. I suspect in the real world the reduction from quad to dual core is something less than 50% as you have certain overhead and activity just running the system w/o BOINC even being up. This will reduce little to none as you change CPUs.

So the following is not based on any good statistics and is much more based on how sluggish the remote GUI become and how often I would detect clients in wait states. All are based on the OC'd headless quads w/ 16 projects running described above. I would probably estimate that you could run at least 3 (or maybe 4 slower) dual core dedicated crunchers in place of 2 of these quads. So maybe use a multiplier of 1.75 for 'medium' dual core machines and 3.0 for 'light' single core machines. But this is all guess work. Adjust by the factor of your choosing.

- **Single Drive for OS and /diskless** – 2 ~ 3 of my 'heavy' crunchers, /diskless partition size 10~15GB.
- **Separate drives for OS and /diskless** – 3 ~ 5 'heavy' crunchers, /diskless drive 20~30GB.
- **Separate drives for OS and RAID 1 for /diskless** – 4 ~ 6 'heavy' crunchers, 2x20~40GB drives (or partitions) in the array. More of a reliability option than anything else.
- **Separate drive for OS and RAID 0 for /diskless** – 7 ~ 14 'heavy' crunchers, 2x40GB drives (for 80GB total), purely a performance option, no redundancy here so consider that.
- **Separate drive for OS and RAID 10 for /diskless** – 7 ~ 14 'heavy' crunchers, 4x40GB drives (for 80GB total), performance & redundancy option.

General RAID stuff -

Being the sorta 'hardware' oriented person that I am I started out I was looking for a hardware RAID solution. I was also looking first at RAID 5 because it looks the best on paper. I quickly found out that to get a RAID 5 controller that had a dedicated processor I was into the 'hundreds of dollars' range. Hardware oriented or not, this was not in the budget. This was a shocker because I'd seen sub \$100 RAID controller cards w/o having checked into what they really could do. It turns out these all use the mobo CPU, rarely support SATAII (that's bound to change) and are often very limited in what they'll handle (2 drives). This started to head me towards just mirroring two drives in RAID 1. Which would resolve my initial concern of keeping the crunchers running on a drive failure.

The more I read the more problems I found folks were having getting their RAID cards to work with various consumer level mobos / chipsets. I never did find a card that was certified to work with Ubuntu or the target mobo I wanted to use. The card I ended up looking at that was 'most likely' to work and supported 3 or 4 drives in RAID 10 or RAID 5 was > \$555. One that might work in RAID 1 w/ a minimalist on-board CPU was around \$165 when I was looking. I'm thinking and this is to hook up 4 drives that I paid less than \$36 for the lot! NOT!

At his point I'm also seeing posts and articles I found while 'googling' that suggested there was very little advantage and some disadvantages to hardware RAID over software RAID on some RAID levels (0, 1, 10). Along with this I'm finding there seems to be some performance issues in the exalted RAID 5 when it's ask to do a lot of small random file writes. It's seems that doing all that building and writing the associated parity blocks has some serious overhead in this situation. And this is just what a diskless client seems to be doing. Now I had to start trying to fathom how software RAID could ever be anything other than old dog slow.

BTW, that RAID 'on the mobo', often called "Fake RAID"... Forget about it if you're running Linux. The drivers are all written for Windows and it's really a fairly ridged way to make the existing SATA controller send some info so software RAID can be implemented in drivers on Windows. Everything I've seen suggests that the Linux kernel level software RAID is a better implementation and a MUCH more flexible way to go.

Well with all this said and looking again at the budget of the moment I figured what the heck, it's free except for my time. Let's see what software RAID can do.

First off you need to chose a RAID level you want.

I'm going to suggest that in most situations RAID 1 (mirrored drives) is your answer. It'll keep things running even if 1 drive fails. It will automatically rebuild itself if one drive has soft errors. It will allow you to replace a truly dead drive with only the downtime to plug it in. Then it will auto rebuild the mirror with the new (or reformatted) drive while it's up and running. It also gives you a small read performance boost and doesn't carry the parity overhead.

This is what I ended up using on my SAMBA sever for both shared network storage and also for the partitions that hold the nightly backups of the other machines (a pair of 500GB Hitachi SATA drives with a 100GB and a 400GB partition on each, the partitions are then mirrored on the other drive). Often RAID 1 is even a good solution for your 'data drive' in your desktop machine. You can find multiple posts, problems and articles about putting the Ubuntu OS onto a RAID'd partition. I personally don't see it worth the trouble. Most of the time it seems to me they are just trying to avoid that plague called “doing backups”.

RAID is **not** a replacement for doing scheduled backups! It might allow doing a backup to the same physical drive (another partition) to make sense though. If you're running any Debian derivative OS you can have sbackup (in Synaptic) up and running scheduled backups in about 15 minutes even allowing for a bit of 'exclude' tweaking. So our excuses are getting thin for not having this in place. Some folks actually rotate out one of the drives in a mirror on a monthly basis and for off-site storage as part of their disaster recovery plan. This typically uses 4 drives, 2 running, 1 local spare, 1 off-site. This is way beyond the scope of anything I need... If my office/garage is burning I'll be way more worried about getting my old car out than saving the cruncher's data.

My personal 'fav' desktop / home server setup is a smallish OS drive with a couple partitions on it (/ , /home and swap) and then a larger drive mounted at /mnt/data. This to me is the drive to create as a single or two partition drive and then mirror those partitions if they contain data that is too painful to restore from backups on a drive failure. Many folks mount /home to a RAID 1 mirrored partition and I can see the logic in this also since I sorta use my 'data' drive as a big /home and use /home for smaller things. I could easily, functionally and physically, combine them to have /home on the RAID partition.

Much good RAID info - [Linux-RAID](#)

General RAID info - [Wiki](#)

For a good definition of RAID 1 or “mirroring” - [pcguide, RAID 1](#)

For a good definition of RAID 10 or a “stripe of mirrors” - [pcguide, RAID 10](#)

While the enclosed diagram in the above link is for RAID 01 (0 + 1, mirror of stripes) he describes how it would look for RAID 10 and does a very good general job of describing various RAID levels.

The whole article in the above links point to is a great read on RAID but it is 80 pages and is likely that it contains more detail than most care about. But then if you got this far in this thing 80 pages on RAID might be right up your alley.

There are a few points I take exception to in that extensive pcguide RAID write-up.

The first is disk costs and size have dropped so much since this was written that the main drawback to RAID 1 and RAID 10 (50% space overhead) is largely a moot point today.

Another is his statements about the write overhead on mirroring. I just don't see it and I'm assuming it's because once the application issues a write request the software RAID (I'm sure this would be implemented as such in hardware also) can mark the request satisfied as soon as the first write done by the hardware. It can satisfy the second write required to the mirror can be done AFTER the write is satisfied to the application. I'm not saying there isn't a write overhead but I think he over-emphasizes it. That extra writing that takes place is load to the i/o bus only but not to the same drive as the first write. Also with even semi-modern drives having 2MB or better of hardware buffering I think you can downgrade the degree of this impact.

Lastly, the entire write up is a bit hardware RAID oriented and someplace in there he makes a comment that software RAID will be "much slower". With modern CPU speeds and HDDs and using a non-parity based software RAID implementation (0, 1, 01, 10) this just isn't true.

Here's a post I found where the guy writing is explaining what/how there could be any advantages to software RAID over hardware raid:

Unlike a hardware solution, if the controller card dies, you can forget about getting your data back since there is no "Standard" for (hardware) RAID. On Linux you could just put the drives into another PC, as the meta-data for software RAID on Linux is not going to change across different versions of Linux.

Examples -

Example /etc/mdadm/mdadm/conf file:

```
# mdadm.conf
#
# Please refer to mdadm.conf(5) for information about this file.
# by default, scan all partitions (/proc/partitions) for MD superblocks.
# alternatively, specify devices to scan, using wildcards if desired.
DEVICE /dev/sdb /dev/sdc /dev/sdd /dev/sde /dev/md0 /dev/md1

# auto-create devices with Debian standard permissions
CREATE owner=root group=disk mode=0660 auto=yes

# automatically tag new arrays as belonging to the local system
HOMEHOST <system>

# instruct the monitoring daemon where to send mail alerts
MAILADDR skip@

# definitions of existing MD arrays
ARRAY /dev/md0 level=raid1 num-devices=2 metadata=00.90
UUID=252431f7:7393664c:180fba5d:ff5e4642
    devices=/dev/sdb,/dev/sdc

ARRAY /dev/md1 level=raid1 num-devices=2 metadata=00.90
UUID=54693081:6615cf45:180fba5d:ff5e4642
    devices=/dev/sdd,/dev/sde

ARRAY /dev/md2 level=raid0 num-devices=2 metadata=00.90
UUID=57bfe363:85826085:180fba5d:ff5e4642
    devices=/dev/md0,/dev/md1
```

boinc@crunch34:/etc/mdadm\$ **sudo fdisk -l**

Disk /dev/sda: 40.0 GB, 40020664320 bytes
255 heads, 63 sectors/track, 4865 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Disk identifier: 0x26aceaed

Device	Boot	Start	End	Blocks	Id	System
/dev/sda1		1	259	2080386	82	Linux swap / Solaris
/dev/sda2	*	260	3523	26218080	83	Linux
/dev/sda3		3524	4865	10779615	5	Extended
/dev/sda5		3524	4865	10779583+	83	Linux

Disk /dev/sdb: 40.0 GB, 40000000000 bytes
255 heads, 63 sectors/track, 4863 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Disk identifier: 0x00000000

Disk /dev/sdb doesn't contain a valid partition table

Disk /dev/sdc: 40.0 GB, 40000000000 bytes
255 heads, 63 sectors/track, 4863 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Disk identifier: 0x00000000

Disk /dev/sdc doesn't contain a valid partition table

Disk /dev/sdd: 40.0 GB, 40000000000 bytes
255 heads, 63 sectors/track, 4863 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Disk identifier: 0x00000000

Disk /dev/sdd doesn't contain a valid partition table

Disk /dev/sde: 40.0 GB, 40000000000 bytes
255 heads, 63 sectors/track, 4863 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Disk identifier: 0x00000000

Disk /dev/sde doesn't contain a valid partition table

If you pre-format the partitions with fdisk and create the array out of the partitions instead of the raw device your fdisk -l output will look more like below. Prettier and I'm sure there must be some advantage to having the partition set to be raid but I haven't found it yet. The final array size will also be a few blocks smaller. On mine it was 39062016 using the partition vs 39062400 using the device.

Disk /dev/sdd: 40.0 GB, 40000000000 bytes
255 heads, 63 sectors/track, 4863 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Disk identifier: 0x00020ff9

Device	Boot	Start	End	Blocks	Id	System
/dev/sdd1		1	4863	39062016	fd	Linux raid autodetect

Disk /dev/sde: 40.0 GB, 40000000000 bytes
255 heads, 63 sectors/track, 4863 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Disk identifier: 0x00020ff9

Device	Boot	Start	End	Blocks	Id	System
/dev/sde1		1	4863	39062016	fd	Linux raid autodetect

```
boinc@crunch34:/etc/mdadm$ cat /proc/mdstat
Personalities : [linear] [multipath] [raid0] [raid1] [raid6] [raid5] [raid4] [raid10]
md2 : active raid0 md0[0] md1[1]
      78124672 blocks 64k chunks

md1 : active raid1 sdd[0] sde[1]
      39062400 blocks [2/2] [UU]

md0 : active raid1 sdb[0] sdc[1]
      39062400 blocks [2/2] [UU]

unused devices: <none>
```

The RAID 1 arrays on my SAMBA server with no activity going on against it:

```
skip@crunch20:~$ sudo hdparm -tT /dev/md0 ; sudo hdparm -tT /dev/md1  
/dev/md0:
```

Timing cached reads: 7990 MB in 2.00 seconds = 3997.16 MB/sec

Timing buffered disk reads: 274 MB in 3.02 seconds = 90.70 MB/sec

```
/dev/md1:
```

Timing cached reads: 7276 MB in 2.00 seconds = 3640.14 MB/sec

Timing buffered disk reads: 264 MB in 3.02 seconds = 87.43 MB/sec

These two arrays are two mirrored partitions on a pair of much newer, faster 500GB Hitachi SATA300 drives.

The 4 drives used in the Dotsch/UX server were much older, slower 40GB Seagate SATA150 drives. But even given that and with a light load on the array the RAID 10's buffered reads are better than the newer hardware RAID 1 array with no load on it. See last read stats on the next page.

Probably more important is the RAID 10 array buffered reads under FULL load is better than then either of its RAID 1 sub-arrays under light load.

I don't have any write stats at this time or any read stats on the RAID 10 array with no load.

I really wish I had discovered hdparm and had these under load stats for all of the configurations I've tried; Single drive, separate drives and the RAID 1 mirror.

The RAID 10 array on my Dotsch/UX disk-less client server while under full load and 10 clients attached:

```
boinc@crunch34:~$ sudo hdparm -tT /dev/md0 ; sudo hdparm -tT /dev/md1 ; sudo hdparm -tT /dev/md2
```

```
/dev/md0:
```

```
Timing cached reads: 3906 MB in 2.00 seconds = 1953.27 MB/sec
```

```
Timing buffered disk reads: 172 MB in 3.01 seconds = 57.20 MB/sec
```

```
/dev/md1:
```

```
Timing cached reads: 3388 MB in 2.00 seconds = 1693.75 MB/sec
```

```
Timing buffered disk reads: 164 MB in 3.05 seconds = 53.73 MB/sec
```

```
/dev/md2:
```

```
Timing cached reads: 3382 MB in 2.00 seconds = 1690.93 MB/sec
```

```
Timing buffered disk reads: 198 MB in 3.34 seconds = 59.21 MB/sec
```

The RAID 10 array on my Dotsch/UX disk-less client server with BOINC stopped on the diskless clients but they are still running:

```
boinc@crunch34:~$ sudo hdparm -tT /dev/md0 ; sudo hdparm -tT /dev/md1 ; sudo hdparm -tT /dev/md2
```

```
/dev/md0:
```

```
Timing cached reads: 5218 MB in 2.00 seconds = 2609.47 MB/sec
```

```
Timing buffered disk reads: 172 MB in 3.01 seconds = 57.08 MB/sec
```

```
/dev/md1:
```

```
Timing cached reads: 5264 MB in 2.00 seconds = 2632.57 MB/sec
```

```
Timing buffered disk reads: 152 MB in 3.02 seconds = 50.30 MB/sec
```

```
/dev/md2:
```

```
Timing cached reads: 5258 MB in 2.00 seconds = 2629.39 MB/sec
```

```
Timing buffered disk reads: 302 MB in 3.01 seconds = 100.38 MB/sec
```