# Building a small NAS using OI

ZFS and OI are well suited to the task of building a small NAS server either for home or small business use. This page documents the process on an HP N40L using OI_151a7, but should generally be applicable to other systems. In fact, if you're building a high end workstation it's very desirable for that also. The RAIDZ arrangement provides significantly faster disk I/O. An alternate title could be "Booting from a RAIDZ Disk Array". That can't actually be done, but this provides as close as you can get right now. 199 MB/s writing 64 GB of /dev/zero to disk is pretty impressive.

## Description of the setup

The HP N40L is a common NAS appliance configuration with room for 4 hard drives in the regular bays and support for ECC memory (up to 8Gb by manufacturer's HCL, though there are reports of 16Gb used with officially unsupported 2x8Gb modules). There is room on the system for an optical drive and an eSATA port, so additional expansion is possible. Some enthusiasts also use 5.25" bay caddies for a number of 2.5" disks – up to 6 disks of the thin 7mm form factor or up to 4 standard thick 12mm disks, usually SSDs or laptop HDDs used for `rpool` (especially with 2Tb+ disks separately used for the storage array), and SSDs for L2ARC and/or ZIL SLOG devices (with an additional HBA as needed). Also a "Russian hacked BIOS" is popular on the Internet (as well as other equivalent tweaks) to allow more flexible use of motherboard's SATA ports on this system.

In my particular instance, the system only has 4x2 TB disks, intended for both the root pool with the OS and for a larger data storage pool. I added a 2 GB ECC DIMM for a total of 4 GB. I did not reflash the BIOS.

At this time, OI cannot boot from EFI (GPT) label disks and SMI (in MBR) label disks are limited to 2 TB. Solaris 11.1 can now boot from EFI drives, but the needed changes have not yet been implemented in OI. The described system will use a small slice on each disk to form a multi-way mirrored root pool and a RAIDZ pool formed from slices that hold all the rest of the space on each disk.

My server is not intended to have a full time console, however, building it is considerably simplified by using the LiveDVD (GUI installer). In particular, this will allow running a disk scan using `format(1m)` on all the drives simultaneously. This takes quite a while for a single disk, so overlapping them by running `format(1m)` in multiple windows speeds things up a good bit. Even then it will take a long time. If you use the text installer, it's more difficult to overlap the media scans. Also, by installing the GUI version you automatically get the packages and libraries required to run a remote VNC console – even if you'd disable the local X11 login minutes after a successful installation. The text-installer shell also suffers from command output scrolling out of view for many of the **format** commands.

> ⚠ **WARNING:** (rhb) The assumption is that there is no data on the drives. If that is not the case you should not follow these instructions (not blindly). These instructions attempt to record all the steps, but make no attempt to instruct a novice user in how to use OI. Even a novice can succeed, but realistically, a complete novice should expect to have to start over a time or two and will need to do lots of additional reading. I had to do it quite a few times to figure how to make it work.

## Scanning the disk media

After the disks are installed in the system, boot from the LiveDVD using either a dedicated optical drive in the optical drive bay or an USB optical drive (or a USB flash with its variant of the installation image).

Open a set of windows, one per disk drive, `su` to `root` and run `format(1m)`.

```
# format -e
```

Select one of the drives (do all the steps in each window once for each drive):

```
format> analyze
```

You'll get a list of options with brief descriptions of what they do. You want to run compare, which writes data to disk, reads it back and compares to see that it's the same. There are a variety of variants of this available. Run:

```
analyze> compare
```

This will take several hours, so don't expect to do anything more for quite a while. For a 2 TB disk, the process takes about 12 hours.

Strictly speaking, this step is not necessary. However, it's going to take quite a while to transfer data to the NAS server after it's been configured, so it makes good sense to make sure all the hardware is in order first (and the long simultaneous scan of all disks also gives a small stress-test on power and cooling, as well, though the system CPU load is quite low).

## Installing

Unfortunately, the current LiveDVD GUI installer will not allow using an existing slice for `rpool`. So it gets a little tricky. By partitioning the disk with `fdisk` to create a small MBR partition we can work around it.

Create a small MBR partition. Then start the installer and select that as the target of the install. Once the install is running take a look at the label with `prtv toc` using a terminal window.

On the other disks create labels with the `s0` slice the same size or larger than the one on the disk that the LiveDVD installer is installing into. Make the `s1` slice all the rest of the disk. Note that the `s2` slice (`backup`) addresses the whole SMI labeled disk, and is best left untouched. However, its existence will force using the "-f" flag for "zpool create". In SunOS 4.1.1 time I (rhb) always eliminated s2, but I find that there are now programs that object strenuously.

> ⚠ **Note about disks with 4KB hardware sector size**
>
> If you are using disks with 4KB hardware sectors, then make sure to use offsets for the start of MBR partition and for start and size of each slice in it that is divisible by 8 units (legacy 512b sector sizes).
>
> The default MBR partitioning starts at 34 or 63 (??) sectors and nether variant does conforms to this advice of 4Kb-alignment, though you might for example start the `s0` slice at an offset of 6 or 1 sectors, as appropriate, into the SMI label to align it with hardware sectors, or – better, cleaner and easier to understand and maintain – use some 256 sectors to offset the MBR partition's start and then 8-factor offsets (including 0) for SMI slices inside the partition. Since the `format` and `fdisk` commands may insist on using "cylinders" of obscure size, you might have more luck setting precise legacy-sector based sizes and offsets with a command-line `parted` tool or its GUI `gparted` sibling.
>
> Also note that the `rpool` created by the installer will, likely, use 512 byte minimum block size (known as `ashift=9`), so the proper alignment will mostly influence the main storage pool that you'd later create manually and have more control over the process.
>
> See the illumos-Wiki page ZFS and Advanced Format disks for more details about this and related issues, as well as possible workarounds to force the OS to use correct `ashift=12` when creating a new pool on your disks – hopefully including the `rpool`.
>
> rhb: The above may or may not be correct. In OI_151_a7 with the ST2000DM001 drives using an SMI label, I could not specify slices except in units of "cylinders" consisting of 255 tracks of 252 sectors of 512 bytes. That makes cylinder 1 not start on a 4k boundary. On my system, the boot slice, s0, runs from cylinder 1-999. There is quite a lot of skew among the various disk utilities about sizes. The lack of alignment seems not to matter, though I spent a lot of time chasing my tail because of it. I would not be surprised to find that what happens is very hardware specific. Attempting to align the start of a slice by editing the output of **prtvtoc** may cause **fmthard** to reject it for not falling on a cylinder boundary.
>
> As of 151_a7, it is not appear to be possible to specify an initial sector which is less than 34.

If your drives are identical, after you've done the first non-install disk, you can copy the label to the others using "`prtvtoc | fmthard -s -`", for example (substitute YOUR drive names as appropriate):

```
# prtvtoc /dev/rdsk/c1t0d0s2 | fmthard -s - /dev/rdsk/c1t1d0s2
```

Verify that all the labels have identical sizes for the `s0` slice or that all are larger than the s0 slice on the install disk. The important point here is that you can attach a larger slice to a mirror, but not a smaller one. Presumably the actual size fo the pool is the size of the smallest slice.

Once the install completes, mirror the install slice onto each of the disks by attaching their `s0` slices to the `rpool`, for example:

```
# zpool attach rpool c1t1d0s0
```

After all the disks have resilvered, you can `reboot` the system. Make sure you boot from the hard drive and not the DVD. The N40L tends to reset to boot device list and order. Other system BIOSes generally behave better/

After the reboot, detach the original install slice from the mirror. If the starting disk was `c1t0d0`, then you'd run:

```
# zpool detach rpool c1t0d0s0
```

Run `fdisk` and increase the MBR to use the entire original disk. If the drives are identical, you can copy the label from one of the other drives using "`prtvtoc |fmthard`". Verify they all match and attach the `s0` slice on the ex-install drive to the `rpool` mirror. If the disks are not identical, go into format and create an s0 slice which is larger than the smallest (??) s0 slice on the disks currently in the active mirror. Then create an `s1` slice using all the rest of the disk for the RAIDZ pool.

Install GRUB on all the bootable disks (components of `rpool`). To install GRUB use a command like this, repeating it for all of your disks:

```
# /sbin/installgrub /boot/grub/stage1 /boot/grub/stage2 /dev/rdsk/c1t1d0s0
```

`reboot` the system again to check that it still does.

Once it comes up, `shutdown` and disconnect the install drive to simulate a failure. If there weren't any mistakes the system will boot properly again, using another disk from the `rpool` mirror.

`zpool status` should tell you that the system is running in "degraded" state. Shutdown, reconnect the install drive and boot – the `rpool` status should be "resilvering" for a while and then "online".

## Next steps

Now you're ready to create a ZFS pool for storage on the larger area of your disks, for my example of a 3-disk raidz1 set:

```
# zpool create tank raidz c1t0d0s1 c1t1d0s1 c1t2d0s1
```

At this time you might want to pay closer attention to the pool properties, such as VDEV layout (for example, you can't remove or add disks to a raidzN set), default checksums, compression, etc. Some of these options you might want to set right into the command used to create the pool.

Also see above for notes about 4KB hardware sectors, if your drives use that. On your main storage pool you especially want to use proper alignment and block sizes. You can use the `zdb` command with no parameters to output your pools' labels and inspect the applied `ashift` value there, recreating the storage pool if needed to fit your configuration.  Be warned though that you'll get the ashift value quickly, but zdb will take a long time to complete on a large filesystem.