UNIVERSITY OF Waterloo

BIF server design and policy

University of Waterloo David R. Cheriton School of Computer Science
University of Waterloo
200 University Avenue West Waterloo, Ontario N2L 3G1 **T** 519-888-4567 x37886 **F** (519) 885-1208 akhudek@cs.uwaterloo.ca

Table of Contents

Common requirements

This chapter lists common requirements for different types of servers. We use **Ubuntu Server** for our operating system except in special instances.

Basic server

All servers must follow these guidelines.

1. One local account for the BIF Administrator with the user name **bifadmin**.
2. The system time is set to UTC.
3. Install the package **postfix** and configure as an "Internet Site."
    1. All system mail goes to **lab-admin@monod.uwaterloo.ca**. Enable this by editing */etc/aliases* and changing the alias for *root* to the above address. Whenever this file is changed, you must run **newaliases**.
4. The message of the day is cleared. Edit both */etc/motd*, and */etc/motd.tail* to do this.
5. Install **libpam-umask** and add the following line to */etc/pam.d/common-session*:

Add to /etc/pam.d/common-session

```
session optional         pam_umask.so umask=066
```

Additional steps if we are running directly on hardware

If we are not running in a virtual machine, we need some additional features such as clock synchronization and RAID.

1. All local file systems must use RAID1, *including swap*.
    1. One large partition for the operating system.
    2. One partition for swap. The swap partition should be *memory_size + 2GB*.
2. Time is managed with **ntpd** using the servers *129.97.128.100*, *129.97.128.10*, and *pool.ntp.org*.

Additional steps to set up a VMWARE host

If a computer is to be a VMWARE host, then use these additional procedures.

1. Only **bifadmin** is allowed to login to the host.

2. Add the Canonical commercial repository to your sources.

Add to /etc/apt/sources.list

```
# canonical commercial repository
deb http://archive.canonical.com/ubuntu feisty-commercial main
```

3. Install the packages **vmware-server** and **vmware-server-kernel-modules**. Also, if not done already add the package **xauth** so that we can run the vmware console over ssh.

4. Restrict access to vmware to root. Normally any user can create a virtual machine. Create the empty file */etc/vmware/create_access* and change the permissions so that only *root* can read and write to it. Add the following to the vmware config file.

Add to /etc/vmware/config

```
serverd.doCreateCheck = "TRUE"
serverd.createCheckFile = "/etc/vmware/create_access"
```

5. There is a bug in the Ubuntu distribution where when starting a virtual machine the *.Xauthority* of file of the user owning the machine is used. This usually means that when using ssh X fowarding, the console can not attach. Since our machines are owned by *root*, we simply create a symbolic link from `/home/bifadmin/.Xauthority` to `/root/.Xauthority`.

6. VMWARE Server 1.0.3 does not support TCP offloading. We disable this on modern cards editing */etc/rc.local/*

Add to /etc/rc.local

```
# VMWARE does not support tcp offloading. This turns it off and allows
# hosts and guests to talk properly.
/usr/sbin/ethtool -K eth0 sg off rx off tx off tso off
/usr/sbin/ethtool -K eth3 sg off rx off tx off tso off
```

Virtual machine guest

Guests require some special steps for proper clock synchronization and performance. These steps must be taken on the host in the guests configuration file.

1. Guest should be set to **Ubuntu** for OS type.

2. Tell vmware to sync time on the guest by adding the following line to the vmx file.

Add or change line in guest.vmx

```
tools.syncTime = “TRUE”
```

The remaining steps are for the guest itself.

1. One local account for the BIF Administrator with the user name **bifadmin**.

2. Install **vmware-tools-kernel-modules** and blacklist the pcnet32 module.

1. To blacklist the pcnet32 module first edit the blacklist file.

   Add to /etc/modprobe.d/blacklist

   ```
   # we use vmxnet and pcnet32 interferes
   blacklist pcnet32
   ```

2. Run **update-initramfs -u**.

3. Tell the kernel not to try to correct time since VMWARE will do it.

   1. First edit menu.lst

      Edit appropriate line in /boot/grub/menu.lst to add clock=pit to default kernel options. Ex:

      ```
      # kopt=root=UUID=daa95329-e62a-4f05-abd5-50ad0d571e44 ro clock=pit
      ```

   2. Run **update-grub** and reboot.

4. Now install the VMWARE tools without any drivers.

   1. Obtain the tools package. This can be copied of the administration server or obtained by selecting to install the tools in the VMWARE console then copying them off the mounted cd.

   2. Run **vmware-install.pl** and accept all the defaults but don't run the configuration.

   3. The configuration will temporarily disable the network, so perform it from the VMWARE console. Run **vmware-config-tools.pl** and select *not* to build any of the drivers. Reboot and make sure the system comes up properly.

Virtual machines

In this chapter we outline the reasoning behind using virtual machines. We try to provide guidance for choosing to create a new virtual machine or to bundle new services into existing virtual machines.

Pros and cons

First we discuss the pros and cons to using virtual machines (VMs).

Better disaster recovery and hardware changes

So long as the VM disk file is on a RAID1 device, downtime is minimized as one can easily just start the image on different hardware in the case of a catastrophic failure. Similarly, if one wants to perform maintenance on a piece of hardware that requires bringing it off-line, it is easy to move the VM to another machine. With XEN, a VM can be moved to another machine *even as it is running*. Finally, moving to new hardware requires a minimal amount of configuration and downtime. It is much easier to configure new hardware as a XEN host rather than replicate the entire existing server. Once the new hardware is configured as a XEN host, an existing machine can be migrated to it with nearly no interruption in service. Network configurations do not need to be changed.

Better security

By partitioning software into VMs we can improve security from a number of standpoints. One, regular users can be partitioned away from critical services thus minimizing accidental interference as well as potential malicious intent. It may be the case that a user account is compromised. By partitioning off critical services, the compromised account has no access to any potential local service exploits, configuration files with incorrect permissions, and it makes it far more difficult to perform denial of service attacks.

Distribution of administration rights

Sometimes people need special web or application servers. In these cases it is best to have the user manage the specialized services. However, if these services run on an existing web server, we do not want the user to be able to inadvertently interfere with the existing service. By giving the user a new VM for their service we can allow the user to have full administration rights over their server. The VM still allows us to control resource usage.

Memory and CPU overhead

Unfortunately VMs impose a memory and CPU overhead. While the CPU overhead may be minimal, the memory overhead can be substantial. This is especially true in the case of virtualized web and database servers. This will be less and less of a problem as hardware improves. However, one must still be cautious of having too many virtual machines.

Administration overhead

While VMs often ease administration, too many VMs can actually cost more time. All these VMs need to be kept up to date with patches. There is currently a lack of good centralized patch management tools for Linux, thus this can become a problem.

Best practice

With the pros and cons of virtual services understood, we now discuss best practice. The big question regarding the use of virtual machines is when to split services into new VMs, or to use VMs at all.

To gain the benefits for disaster recovery and hardware changes we want as many services in VMs as possible. There are a few cases where this is either not possible or not a good idea. To answer the above question, we give a number of questions that you should consider.

1. Does the CPU overhead of the VM matter? I will use the example of a cluster node for high performance computer. This is both a good example and extremely relevant for BIF. The point of a cluster node is to provide raw CPU power and memory. The CPU and memory overhead of a VM therefore becomes a problem. Also, there are often numerous cluster nodes and if one fails, its resources can not be migrated via a VM. Therefore the benefits of disaster recovery and hardware changes do not apply. We conclude that *VMs are not appropriate for cluster nodes*.

2. Do the services require administration rights to be delegated to someone? If this is the case, then it may be a good idea to put the service in a new VM.

3. Is a particular collection of services more of a security risk than others? For instance, services that are both complex and provide services both to the public and locally may be at higher risk for exploit. In the case of BIF, the main web server, imap, and smpt services are both complex and publicly broadcast. They will therefore get a lot more attention from people looking to break into our systems. Also, being complex, the possibility for an exploit is much greater. Therefore we partition these services into a separate VM and even use a separate password for the admin account.

4. Otherwise, we should bundle services into one VM to reduce overhead and administration.

Choosing virtualization software

There are several virtual machine offerings for Linux. Since this is for production use, we can limit our evaluation to the largest and most tested solutions. Despite *KVM* now being supported by Ubuntu directly, reports from the internet show that it is not close to ready for use in a production environment. There is a lot of talk about a lack of stability and big problems with performance. Also, *KVM* requires very modern CPU support and as such will not work on our older computers. This leaves us with two systems: *XEN* and *VMWARE Server*.

XEN on Ubuntu

XEN is open source and has very nice features that include nearly live migration and resource throttling. I evaluated this software in our environment and found numerous problems.

1. Requires special kernel patches. The XEN Ubuntu kernels are not directly supported by Ubuntu and seem to be quite a bit behind the normal Ubuntu kernels in releases. This is a big deal as a good kernel is essential to a stable system. Running an unofficial kernel that lags in tracking the main kernel is a terrible idea. We've already had problems with this while running Mosix.

2. Running Windows or other operating systems require modern CPU support. Not a big deal, but it would be nice to have this functionality.

3. Stability problems. During testing I had guest operating systems behaving strangely. While working with network settings I encountered a kernel panic that brought down the whole system. This is unacceptable.

Because of these points, I chose to use *VMWARE*.

VMWARE on Ubuntu

VMWARE has a free offering called VMWARE Server. I will cover both the pros and cons of this free offering.

Pros

1. It is tried and true. VMWARE is already used in production environments all over the world.
2. It can run on both modern CPUs and older CPUs. The guest OS is not dependant on a particular host OS. We can even run our Linux servers on a Windows host if we wanted to. While we wouldn't, this flexibility is good.
3. It has the possibility of commercial support as well as more complex and powerful paid products that we can upgrade too.
4. Canonical has VMWARE packages that track the latest Ubuntu kernel. This makes upgrades easy.

Cons

1. It is not open source. The free version can go away at some point. Hopefully by then XEN will have matured enough to provide an alternative. It is possible to convert VMWARE guest images to XEN guest images.
2. Guests only support up to two processors in the free version.
3. No resource limiting in the free version.
4. No virtual machine migration in the free version.
5. Poor documentation, especially when it comes to limiting access in a Unix environment.

Conclusion

During testing we found that XEN is not ready for production use on Ubuntu. We would like to see official Ubuntu XEN kernels and a lot more testing before we reconsider XEN. This forces us to go with the free VMWARE Server.

Hosting virtual images

We have two options for hosting virtual images: on local storage, or on an NFS drive. The advantage of using local storage is better performance. On the other hand, a critical hardware failure requires extracting the drives and copying the image over to new hardware. Precisely this happened when transcriptome.cs.uwaterloo.ca failed and we had to move the monod virtual image to new hardware. While our setup performed reasonably, reducing down time to half a day, we were surprised by the time it took to extract the drives and copy over the virtual image.

The down time could be further reduced by using NFS backed storage for our virtual images. Additionally, by storing our images on ZFS through NFS, we can make use of ZFS snapshots to save consistent images of our servers. This would have been a huge help when dna.cs.uwaterloo.ca was hacked in 2009. Rather than rebuilding the entire server, snapshots would have allowed us to simply roll the server back to before it was hacked and immediately patch the exploit that was used to gain access to the system.

For these reasons, we currently run dna.cs.uwaterloo.ca off of NFS. Although performance is not as good, it seems to work fairly well so far. We might consider moving our other images onto NFS in the future. The only concern is the added load we put on the file server in terms of available read and write bandwidth.

Solaris native LDAP client

This is a partial HOWTO on getting the Solaris 10 native ldap client working with an openldap server. We assume the reader is already familiar with OpenLDAP and has a working installation. We will also attempt to configure the Solaris client to use TLS/SSL.

Requirements

- OpenLDAP 2.3.xx
- OpenSSL (needed for TLS)
- DUAConfigProfile.schema
- solaris.schema
- server cert, server key
- ca cert

Walkthrough

With the above ready, we proceed with the walkthrough. *Note, with the latest ubuntu the Solaris ssl library can no longer talk to the Linux ssl library. Thus we are currently using unencrypted LDAP over the private network.*

1. First we configure **slapd**. There are two important steps here. We must include the new schema files, and to allow TLS, bind **slapd** to *both* ports 389 and 636 (ldap and ldaps). It appears the Solaris native client does not support start_tls, but does support SSL. Even so, there are reports that ssl is used for authentication only. Further, if you must allow unencrypted access.
2. Next we must configure the directory itself.
    1. Top level entry. We need to add the object class nisDomainObject and nisDomain property.

Modify top level object.

```
dn: dc=bioinformatics,dc=uwaterloo,dc=ca
o: bif
dc: bioinformatics
nisDomain: bioinformatics.uwaterloo.ca
objectClass: dcObject
objectClass: nisDomainObject
objectClass: organization
objectClass: top
```

2. Next we add a profile group and default profile. If you do not want TLS/SSL, then change authentication method from *tls:simple* to *simple.*

Add profile.

```
dn: ou=profile,dc=example,dc=com
objectClass: top
objectClass: organizationalUnit
ou: profile

dn: cn=default,ou=profile,dc=example,dc=com
objectClass: top
objectClass: DUAConfigProfile
defaultServerList: ldapserver
defaultSearchBase: dc=example,dc=com
defaultSearchScope: one
searchTimeLimit: 30
bindTimeLimit: 2
profileTTL: 43200
followReferrals: FALSE
cn: default
authenticationMethod: tls:simple
credentialLevel: proxy
serviceSearchDescriptor: passwd: ou=People,dc=example,dc=com?one
serviceSearchDescriptor: group: ou=Group,dc=example,dc=com?one
serviceSearchDescriptor: shadow: ou=People,dc=example,dc=com?one
```

3. Finally, we add a proxyagent. The proxy agent is the bind name that the solaris client uses to access the directory. The password for the proxy agent *must* be in crypt format. The solaris client will show a password of type NS1 in it's configuration. This is just a simple form of encryption to avoid storing the password in plain text. The solaris client will decrypt this and then authenticate using crypt.

Add profile.

```
dn: cn=proxyagent,ou=profile,dc=bioinformatics,dc=uwaterloo,dc=ca
cn: proxyagent
sn: proxyagent
objectClass: top
objectClass: person
userPassword: {CRYPT}cryptedpassword
```

3. Next we configure the client. First let us create our key database. For this we use *certutil* from the openssl package. First place your server.pem, key.pem, and ca.crt in /var/ldap.

In /var/ldap

```
# create a new database
$ certutil -d . -N

# add the CA and assign it to be trusted
$ certutil -d . -A -n myca -t "C,C,C" -i ca.crt

# add both the cert and key to a single file
$ cat server.pem key.pem > both.pem

# convert to p12 format
openssl pkcs12 -export -in both.pem -out both.p12 -name ldapserver

# add the client cert and key
$ pk12util -d . -i both.p12 -K

# check that things went ok
$ certutil -d . -L
ldapserver u,u,u
myca C,C,C

$ certutil -d . -V -n ldapserver -u V
certutil: certificate is valid
```

Make sure that cert8.db, cert7,db and secmod.db are mode 644. You can remove the starting files.

4. Edit /etc/nsswitch.ldap ... (ipnodes files only, hosts files first, etc)

5. Edit /etc/defaultdomain

In /etc/defaultdomain

```
bioinformatics.uwaterloo.ca
```

You can also set the domain without rebooting with the command

Set domainname without reboot

```
domainname bioinformatics.uwaterloo.ca
```

6. Finally we initialize LDAP with ldapclient.

Initialize LDAP

```
# ldapclient -v init \
-a proxyDN=cn=proxyagent,ou=profile,dc=bioinformatics,dc=uwaterloo,dc=ca \
 -a proxyPassword=<proxypassword> evolve-local

Parsing proxyDN=cn=proxyagent,ou=profile,dc=bioinformatics,dc=uwaterloo,dc=ca
...
restart: sleep 100000 microseconds
restart: milestone/name-services:default... success
System successfully configured
```

Managing systems

Network layout

All hosts are dual homed. One interface is for a local network, the other for the university network. We assign the fully qualified domain name to the university interface, and "*hostname*-local" to the private interface. We manage names through local host files which we keep in sync with the puppet configuration manager.

For speed and security, LDAP and NFS use the private interface. The public interface is for internet access and administration access.

Services

The cluster offers many services which we summarize below. Our primary and secondary file servers are `hmsbarracouta` and `hmsbeagle` respectively. These run Solaris with ZFS while the remaining machines run Ubuntu server. Our web and printer server is `monod`. Administration services such as LDAP, the grid engine master, and the puppet master, run on `evolve`. Users only have access to `dna` which we intend as a general purpose compute node and primary access point to all other services. Some machines are virtual, running on the current host, `codon`.

Machine	Service	Service name	Dependancies	Notes
monod (virtual)	http	apache	nfs, ldap	
	smtp	postfix	nfs, ldap	
	imap	N/A		
	cups	cupsys		
	puppet client	puppet	puppetmaster	
evolve (virtual)	ldap	slapd		
	cplex license manager	ilog-ilm		
	puppet master	puppetmaster		
	grid engine master	sgemaster.uwbif	nfs	
	puppet client	puppet	puppetmaster	
hmsbarracouta	nfs			primary
	samba			primary

Machine	Service	Service name	Dependancies	Notes
hmsbeagle	nfs			backup
	samba			backup
dna (virtual)	ssh		nfs, ldap	All machines have ssh, but regular users can only log in to dna.
	puppet client	puppet	puppetmaster	
codon	grid engine exec	sgeexecd.uwbif	nfs, ldap	
	vmware web console	vmware	nfs	https://codon.cs:8333/
	puppet client	puppet	puppetmaster	
chromosome	grid engine exec	sgeexecd.uwbif	nfs, ldap	
	puppet client	puppet	puppetmaster	
histone	grid engine exec	sgeexecd.uwbif	nfs, ldap	
	puppet client	puppet	puppetmaster	
chromatin	grid engine exec	sgeexecd.uwbif	nfs, ldap	
	puppet client	puppet	puppetmaster	

Configuration management with puppet

We manage as much as possible through puppet. This includes all installed packages and all configuration files. This ensures consistency while still having complex security configurations and allows for rapid installation of new servers. For a full description of how puppet works, please see the puppet website: http://reductivelabs.com/products/puppet/. In this section we limit our description to the unique aspects of our puppet configuration.

Configuration files

Our puppet configuration files are on `evolve` in the `/etc/puppet` directory. The majority of the configuration logic is in the `/etc/puppet/modules` directory. We follow standard puppet module procedures.

Module	Purpose
acct	Process accounting.
apt	`apt` config files.
bash	`bash` config files and associated default profiles.
bifaccounttools	Dependancies for BIF password change and user addition tools.
cluster	Various cluster configurations including package installation list for compute nodes and interactive nodes.
cron	`cron` entries that do not fit anywhere else.

Module	Purpose
csh	`csh` package and related configuration files.
cups	Printer client files.
ldap	LDAP client and server configuration.
motd	Message of the day.
net	NFS mounts.
network	Configuration for interfaces and DNS.
nfs	NFS services and configs.
ntp	Time management configuration through `ntp`.
pam	Authentication services. This includes modifications for LDAP as well as limiting user logins to interactive nodes.
postfix	Default mail services for all machines.
puppet	Mostly client configuration. Puppet can manage itself if you are careful.
security	Limits and access configuration to go with the PAM configuration files.
ssh	SSH services.
tcsh	`tcsh` and related configuration files.
user	Local users such as bifadmin, cscf-admin, and database admin.

The remaining configuration is in `/etc/puppet/manifests`. Here, `templates.pp` provides basic templates for various types of machines including cluster nodes, interactive nodes, servers, and Solaris machines. We configures machines themselves in the `nodes.pp` file. The `site.pp` file only loads our templates and nodes at this point.

Known problems and fixes

Puppet clients seem to freeze or break occasionally. We use a cron entry to restart the puppet client every two hours, if it is stopped. To find frozen clients, we run `puppet_check` on evolve. This script looks at the puppet master database and emails a warning for any clients that have node checked in within a day.

Puppet's management of Solaris is currently somewhat limited and thus we manually manage our Solaris machines for the most part. Initially we managed cron through puppet, but it turns out that you can only install root cron entries on Solaris machines.

Accounts

We use both system level accounts and accounts managed through LDAP. We use LDAP for all user accounts and system level accounts for the rest. The reasoning is that administration and service accounts may need to operate when the LDAP server is offline. We synchronize system accounts with puppet and provide custom scripts to manage LDAP users.

System accounts

The table below lists all system level accounts managed through puppet. The zfsadmin account is only present on hmsbeagle and hmsbarracouta.

Account	Purpose
bifadmin	Administrator account with access to root through sudo.
cscf-admin	Administrator account with access to root through sudo.
dbadmin	To synchronize the local BLAST databases on each cluster node.
zfsadmin	To synchronize the two file servers. To facilitate automated backup operations via a cron script, this account uses a shared RSA key on hmsbeagle and hmsbarracouta.

User accounts

All user accounts are managed through LDAP. Login access to the different machines is controlled by user groups. Active users must be part of the `research` group and are currently allowed to log in to dna.cs.uwaterloo.ca only. Administrators are pare of the `admin` group and can log in anywhere. The intention behind this setup is to allow for finer control over users. For example, by adding a group and adjusting other configuration settings we can put different limits on alumni.

Creating user accounts

Creating a user account involves making an LDAP entry with appropriate password and creating a home directory through ZFS. The home directory must be created as a ZFS file system. This entire process is automated, including generation of a random password, by a custom program called `bif-adduser`. This program will create the user as part of the `research` group.

Example user creation

```
root@evolve:~# bif-adduser testuser
Enter ldap password:
Creating user testuser with a uid and gid of 2121.
Full name : Test User
Office    : DC1234
Phone     : 1-519-123-4567
Current email testuser@cs.uwaterloo.ca (leave blank for default)
Email     :
Purpose   : Test User
Date created 2010-02-09.
ex. browndg, mli
Supervisor: nobody
Generated password: ucsyiderg
Created group.
Created user.
Added to research group.
Created home directory.
root@evolve:~#
```

Known bug: The password field can have problems with extreme characters such as unicode. I believe backslashes may cause problems as well.

Deleting user accounts

We have no policy on deleting user accounts at this time. If you make an error in creating an account and really need to delete it, you must do so manually. First, use `ldapvi` as `bifadmin` on evolve and delete all entries related to the user. There should be a group, a user, and an entry in the research group. You may use another LDAP editor if you prefer. Second, you must log into `hmsbarracouta` and `hmsbeagle` and delete the users ZFS home directory using `zfs destroy`. If you have just created the user, only `hmsbarracouta` will have an account as the backup task will not yet have backed up the new directory.

Changing user passwords

To change a user's password in LDAP, they must use the `bif-passwd` program. This program will change both the unix and samba password hashes. The default shell configuration for both bash and csh/tcsh is currently set to alias `passwd` to `bif-passwd`. If this alias is corrupted for any reason, it will result in difficulty changing passwords. The `bif-password` program is subject to the same special character bug as `bif-adduser`.

Filesystem and backup

Our filesystem and backup system has several requirements: snapshots, mirrors backup to a second machine, tolerance of a single drive failure, tolerance of a catastrophic failure of a critical server component. We describe how our current system meets all these needs.

Architecture

Since tape systems are fairly expensive, we chose to use a disk based backup system. Our basic architecture is a pair of identical file server. Each file server uses SATA RAID and can tolerate a single drive failure with no down time. All drives are hot-swappable. Having two identical file servers also gives us a solution for the situation in which we have a catastrophic hardware failure on the primary file server. In that case, we can swap in our backup file server as a temporary primary.

Solaris and ZFS

We chose Solaris with ZFS as the operating system for our file server. Although we try to standardize on a single platform, the benefits of Solaris and ZFS outweigh the downside of having to support two different operating systems. ZFS provides several critical features: snapshots, incremental cross-system backup, compression, and flexible software configuration.

File system configuration

We export the file system with NFS3. Although ZFS supports NFS4, in testing we found that the Linux NFS4 client caused occasional kernel panics. Similarly, the Linux NFS client does not support NFS ACLs, although Solaris and ZFS supports this too. These features should be experimented with again when Linux is upgraded. NFS4 gives substantial performance and safety improvements and ACLs would eliminate the need to create custom unix groups to allow groups of researchers to share project files.

We make use of ZFS's light-weight file systems by creating every user directory as a separate file system. This design allows for per-user quotas as well as other possibilities. We also enable compression on all user directories. Since bioinformatics data is largely text-based, compression gives nearly 50% space saving. Additionally, in modern times compression actually improves performance rather than detracting from it. CPUs today have no problem handling the small computational load needed to compress and uncompress data. File system performance is largely limited by physical reads and writes and compression reduces both numbers.

Snapshots and backup sync

Snapshots are critical to our backup infrastructure. Our current policy is to keep one weekly snapshot, and the last three daily snapshots. Only daily snapshots are synchronized with the backup file server. It is important to note that to perform incremental backups, we must always keep the last synchronized snapshot. All snapshots and remote syncs are managed by the `/export/home/zfsadmin/backup_manager.pl` program. Daily snapshots use the backup manager through the `/export/home/zfsadmin/daily.sh` script and weekly snapshots through the `/export/home/zfsadmin/weekly.sh` script. Both these latter scripts are called through `zfsadmin`'s crontab. The backup manager program not delete the last snapshot shared between hmsbarracouta and hmsbeagle.

RAID, drive layout, and performance

Both our file servers are equipped with true hardware SATA RAID. Currently, hmsbarracouta uses four drives in a RAID5 configuration for all data, and two drives in a RAID1 configuration for the operating system. This should make upgrades of either the drives or operating system fairly easy. The backup file server, hmsbeagle, uses the same layout, but uses ZFS's ZRAID1 for the data partition instead of hardware RAID. This change was made for two reasons: ZRAID is supposedly safer than RAID5 because it addresses the RAID5 write hole, and we were testing to see if software RAID is faster than our hardware RAID.

Our concerns with speed come from underwhelming read and write performance by our file servers. Sadly, there was no indication that the hardware would under-perform in any way when we bought it. However, write performance in particular is very poor with both hardware and software RAID setups. We average 14MB/s on write. Read performance is at an acceptable 60MB/s. One possible reason is a slow SATA bus. Extensive testing and research did not result in any obvious hardware or software failure or choke points.