**Starting up NFS**

To better explain the steps for configuring NFS on an existing server, I’ll use a server that I’ve used in past articles. The highlights of the server test system used in this article are:

* CentOS 5.4
* GigaByte MAA78GM-US2H motherboard
* An AMD Phenom II X4 920 CPU
* 8GB of memory (DDR2-800)
* Linux 2.6.30 kernel (with reiser4 patches only)
* The OS and boot drive are on an IBM DTLA-307020 (20GB drive at Ultra ATA/100)
* /home is on a Seagate ST1360827AS
* There are two drives for testing. They are Seagate ST3500641AS-RK with 16 MB cache each. These are /dev/sdb and /dev/sdc.

Only the first Seagate drive was used, /dev/sdb, for the file system, which in this case is ext4. The second hard drive, /dev/sdc was used for the journal portion of the file system. It was partitioned to the correct size and only that partition was used for the journal (/dev/sdc1).

The first step in creating a NAS box from an existing server is to simply configure and start up NFS. I will assume that you have installed NFS on your system including the NFS server component (it is beyond the scope of this article to walk through the various distributions to explain how to do it). One way to check if NFS is installed on the server is if the file /etc/exports exists (it may not have anything in the file, but the file should exist). In addition, on CentOS or RHEL, you can also check if the NFS server is installed by looking for the file /usr/sbin/rpc.nfsd.

**[laytonjb@test64 ]$ ls -s /usr/sbin/rpc.nfsd**

**20 /usr/sbin/rpc.nfsd**

For rpm based systems you might also try the following:

**[laytonjb@home8 ]$ rpm -qa | grep -i nfs**

**system-config-nfs-1.2.8-1**

**nfs-utils-1.0.6-46**

(this is an old CentOS 4.1 system so the versions numbers will definitely not match anything newer). At this point let’s assume that the server components of NFS are installed.

Typically I next start NFS on the server. A quick way to check if NFS is running is to use the command, “rpcinfo -p” which reports the RPC (remote procedure call) information on the system. If NFS server has not been started, you will see something like the following.

**[root@test64 ]# /usr/sbin/rpcinfo -p**

**program vers proto port**

**100000 2 tcp 111 portmapper**

**100000 2 udp 111 portmapper**

**100024 1 udp 774 status**

**100024 1 tcp 777 status**

The most important thing in this output is that the “portmapper” is running which is an important part of NFS. If yout don’t see [portmapper](http://en.wikipedia.org/wiki/Portmap) in the list, please read your documentation on how to get it running.

To start the NFS server, or really the server portion of NFS, depends upon your distribution to some degree. On the test CentOS system it was simply accomplished by the following command.

**[root@test64 ]# /etc/init.d/nfs start**

We then run “rpcinfo -p” again to see if NFS was started.

**[root@test64 ]# /usr/sbin/rpcinfo -p**

**program vers proto port**

**100000 2 tcp 111 portmapper**

**100000 2 udp 111 portmapper**

**100024 1 udp 774 status**

**100024 1 tcp 777 status**

**100011 1 udp 645 rquotad**

**100011 2 udp 645 rquotad**

**100011 1 tcp 648 rquotad**

**100011 2 tcp 648 rquotad**

**100003 2 udp 2049 nfs**

**100003 3 udp 2049 nfs**

**100003 4 udp 2049 nfs**

**100021 1 udp 42049 nlockmgr**

**100021 3 udp 42049 nlockmgr**

**100021 4 udp 42049 nlockmgr**

**100021 1 tcp 54256 nlockmgr**

**100021 3 tcp 54256 nlockmgr**

**100021 4 tcp 54256 nlockmgr**

**100003 2 tcp 2049 nfs**

**100003 3 tcp 2049 nfs**

**100003 4 tcp 2049 nfs**

**100005 1 udp 669 mountd**

**100005 1 tcp 672 mountd**

**100005 2 udp 669 mountd**

**100005 2 tcp 672 mountd**

**100005 3 udp 669 mountd**

**100005 3 tcp 672 mountd**

Notice that you see NFS listed and that NFS v2, NFS v3, and NFS v4 were started (look at the second column). Also notice that the UDP and TCP protocols for NFS are listed as well. The program mountd, is the NFS mount daemon.

There are a number of daemons needed for NFS to operate. The /etc/init.d/nfs script started everything for us but in general the [daemons we need](http://www.linux.org/docs/ldp/howto/NFS-HOWTO/server.html) are:

* rpc.nfsd (server)
* rpc.lockd (common to both server and client)
* rpc.statd (common to both server and client)
* rpc.mountd (common to both server and client)
* rpc.rquotad (server)

**Exporting Data**

The next step is to tell the NFS server what directories can be shared with other systems (clients). In the vocabulary of NFS, this is called “exporting” directories and the specific directories are called “exports.” Now that NFS is started, let’s configure it to export a directory on the server.

To do this, we edit the file /etc/exports by defining the directories to be exported and their properties. The typical entry in /etc/exports looks like the following:

**directory machine1(option11,option12) machine2(option21,option22)**

where:

* **directory** is the server directory you want to export to clients. It can be a single directory or an entire drive. However, it has to be formatted with a file system. Each line in /etc/exports is a separate directory.
* **machine1, machine2** are the names of clients you wish to share data with. For example they could be listed using system names such as, client1 or client2 (be sure the clients are listed in /etc/hosts or via NIS), or they could be listed by IP address such as 192.168.1.8. You can also used the wild card, “\*”, to indicate any client. It is **HIGHLY**recommended that you list every client here to help ensure that no “rogue” clients can mount the exported directory. It is a pain to maintain the list of clients, particularly if it is a long list, but this little bit of security can help with unsophisticated attacks. On the other hand, if you are behind a big juicy firewall and are confident in it’s abilities and that no one will cause any hard either intentional or by accident, then you can just list the machines as “\*” which means all clients (many times HPC clusters will do this because the cluster is on a private network which is usually behind a big juicy firewall).
* **(option11, options12, …)** are a list of options used for exporting the directory. There are a number of options that can be used and it’s beyond the scope of this article to present then all. However, some of the more important ones are:
  + **ro** which stands for read-only. So the server can export directories that are read-only so the clients cannot write to them.
  + **rw** which stands for read-write. This means that the clients can read and write to the exported directory.
  + **no\_root\_squash** which means that user “root” on the client machine will have the same level of access to the files on the system as user “root” on the server. Again, there are a number of security issues surrounding this and it is recommended that you do NOT use no\_root\_squash on the clients.
  + **sync** which tells NFS to wait until the data has been flushed to the storage device before returning. The other option is async which allows the NFS server to return to the client application before the data has been sent to the device (i.e. it may just be in cache somewhere). It is recommended that you use the “sync” option to ensure that the data has been written to permanent storage. However, there is a performance penalty for this option with NFS file systems mounted using the “sync” option being slower than “async.”

There are many ways in which you can export directories to clients. It all depends upon what you want to accomplish and how you want to accomplish it. For example, you could export a directory on the server that contains applications that the clients might need as shown below in a sample line from /etc/exports.

**/opt 192.168.1.8(ro)**

In this case the server is exporting (sharing) the directory, /opt which contains some applications, to a client, 192.168.1.8, which can mount it read-only (no writing). This is a fairly common way of installing applications on a single server and sharing them with other clients.

One of the biggest uses of NFS is for home directories. It is fairly easy and common to put user’s home directories on a single server and then mount it on the clients. Such an entry in /etc/exports might look like the following:

**/home 192.168.1.8(rw)**

In this case, /home is mounted as read-write on the client 192.168.1.8 since we want users to be able to write to their own home directory. If you want, you can get more granular and specify each user on a line in /etc/exports and control which clients that user will have a home directory. This gives you some control at the expense of extra work on the NFS server along with some additional complexities. For example, you might want to do something such as the following.

**/home/laytonjb 192.168.1.8(rw)**

**/home/test 192.168.1.65(rw)**

In this case the first user, laytonjb, is exported to client 192.168.1.8 and the second user, test, is exported to client 192.168.1.65. This gives you fairly fine control over what is exported and to which machines it is exported. Coupling this with a fairly good way to update /etc/fstab on the clients gives the administrator great control over data access.

Again, NFS security is not a focus of this article, but to help yourself, it is recommended that you use the files /etc/hosts.allow and /etc/hosts.deny. They are strictly not necessary but it does give the administrator a bit more control over the NFS configuration. The first file can be used to define which clients are allowed to use services on the host machine and the second file lists which clients are denied access to certain services on the host. While this [link](http://www.linux.org/docs/ldp/howto/NFS-HOWTO/server.html)is a bit old, it does explain how to use these files to better protect your system.

For the simple example in this article I have just made a single entry in /etc/exports.

**/mnt/home1 192.168.1.8(rw)**

It exports a directory called /home1 to a client, 192.168.1.8, and allows read and write access to it. After I save the file I have to tell NFS to re-read the /etc/exports file and re-export it (i.e. make it available to the client). The command for this is simple.

**[root@test64 ]# /usr/sbin/exportfs -ra**

The command exportfs controls the list of exported directories and the options “-ra” tell it to export all directories listed in /etc/exports (the “-a” option) and to “re-export” all directories in /etc/exports (the “-r” option).

**NFS Client**

The server side of NFS is done with at least one directory, in this case /mnt/home, exported to a client on my network. The client side of NFS is fairly easy as well. As with the server you have to have NFS installed on the client but you don’t have to have the server portion of NFS installed. On many distributions the “client” piece of NFS is sometimes called “nfs-common” or sometimes “nfs-client” or something similar. A generic way to check if your system is NFS ready is to look at /proc/filesystem.

**laytonjb@laytonjb-laptop:~$ cat /proc/filesystems**

**nodev sysfs**

**nodev rootfs**

**nodev bdev**

**nodev proc**

**nodev cgroup**

**nodev cpuset**

**nodev debugfs**

**nodev securityfs**

**nodev sockfs**

**nodev pipefs**

**nodev anon\_inodefs**

**nodev tmpfs**

**nodev inotifyfs**

**nodev devpts**

**cramfs**

**nodev ramfs**

**nodev hugetlbfs**

**nodev mqueue**

**nodev fuse**

**fuseblk**

**nodev fusectl**

**nodev usbfs**

**ext3**

**nodev rpc\_pipefs**

**nodev nfsd**

**nodev binfmt\_misc**

The second line from the bottom indicates that this system is indeed capable of NFS.

As with the server you need several daemons for running NFS on the client. In particular you need three daemons for the client:

* portmapper (use “rpcinfo -p” to check)
* rpc.statd (needed for file locking)
* rpc.lockd (needed for file locking as well)

These should be started at boot when you installed the NFS client portion.

The final step is to simply mount the NFS exported directory. If you want to test it by hand you can use “mount” on the command line.

**[root@home8 etc]# mount 192.168.1.65:/mnt/home1 /mnt/nfsserver**

**[root@home8 etc]# ls -s /mnt/nfsserver**

**total 8**

**4 laytonjb 4 test**

**[root@home8 etc]# ls -s /mnt/nfsserver/laytonjb**

**total 40**

**20 ext4\_own\_journal.txt 4 fdtest\_script 16 fdtree.bash**

where the NFS server is 192.168.1.65. Be sure the mount point, /mnt/nfsserver exists before you mount the file system or you will get an error message that the mount point doesn’t exist.

You can also put this in your /etc/fstab file. An example /etc/fstab file looks like the following.

**[root@home8 etc]# more /etc/fstab**

**# This file is edited by fstab-sync - see 'man fstab-sync' for details**

**LABEL=/ / ext3 defaults 1 1**

**LABEL=/boot /boot ext2 defaults 1 2**

**none /dev/pts devpts gid=5,mode=620 0 0**

**none /dev/shm tmpfs defaults 0 0**

**none /proc proc defaults 0 0**

**none /sys sysfs defaults 0 0**

**LABEL=SWAP-hda3 swap swap defaults 0 0**

**...**

**192.168.1.65:/mnt/home1 /mnt/nfsserver nfs defaults 0 0**

I have dropped some junk from the /etc/fstab file to protect the innocent (me). Also the client is an older CentOS 4.1 system so the /etc/fstab file may not match what you have on your system. However, the syntax for the entry in the file is correct.

**Summary**

Taking an existing server and turning it into a NAS box that functions as a central server for a set of clients is actually fairly simple. You can take your favorite distribution, make sure NFS and NFS Server are installed, and configure NFS on the server. Then on the clients you configure each client to mount the exported file systems from the server. The process is not difficult, but perhaps a little time consuming.

This article has just briefly touched on the subject of security for NFS. This can be an important issue if you are worried about possible security issues and you should be if you are operating on a network with clients (HPC is another story for a different article). Be sure to use the many web based articles and HOWTO’s around NFS security. There are also security books that talk about securing NFS (my personal favorite is “[Real World Security](http://www.amazon.com/Real-World-Linux-Security-2nd/dp/0130464562/ref=sr_1_1?ie=UTF8&s=books&qid=1273666460&sr=8-1)” by Bob Toxen. It’s perhaps getting a bit old, but the general discussion around NFS is quite good)