NAME

pNFSserver - NFS Version 4.1 and 4.2 Parallel NFS Protocol Server

DESCRIPTION

A set of FreeBSD servers may be configured to provide a pnfs(4) service. One FreeBSD system needs to be configured as a MetaData Server (MDS) and at least one additional FreeBSD system needs to be configured as one or more Data Servers (DS)s.

These FreeBSD systems are configured to be NFSv4.1 and NFSv4.2 servers, see nfsd(8) and exports(5) if you are not familiar with configuring a NFSv4.n server. All DS(s) and the MDS should support NFSv4.2 as well as NFSv4.1. Mixing an MDS that supports NFSv4.2 with any DS(s) that do not support NFSv4.2 will not work correctly. As such, all DS(s) must be upgraded from FreeBSD 12 to FreeBSD 13 before upgrading the MDS.

DS server configuration

The DS(s) need to be configured as NFSv4.1 and NFSv4.2 server(s), with a top level exported directory used for storage of data files. This directory must be owned by "root" and would normally have a mode of "700". Within this directory there needs to be additional directories named ds0,...,dsN (where N is 19 by default) also owned by "root" with mode "700". These are the directories where the data files are stored. The following command can be run by root when in the top level exported directory to create these subdirectories.

```
jot -w ds 20 0 | xargs mkdir -m 700
```

Note that "20" is the default and can be set to a larger value on the MDS as shown below.

The top level exported directory used for storage of data files must be exported to the MDS with the "maproot=root sec=sys" export options so that the MDS can create entries in these subdirectories. It must also be exported to all pNFS aware clients, but these clients do not require the "maproot=root" export option and this directory should be exported to them with the same options as used by the MDS to export file system(s) to the clients.

It is possible to have multiple DSs on the same FreeBSD system, but each of these DSs must have a separate top level exported directory used for storage of data files and each of these DSs must be mountable via a separate IP address. Alias addresses can be set on the DS server system for a network interface via ifconfig(8) to create these different IP addresses. Multiple DSs on the same server may be useful when data for different file systems on the MDS are being stored on different file system volumes on the FreeBSD DS system.

MDS server configuration

The MDS must be a separate FreeBSD system from the FreeBSD DS system(s) and NFS clients. It is configured as a NFSv4.1 and NFSv4.2 server with file system(s) exported to clients. However, the "-p" command line argument for nfsd is used to indicate that it is running as the MDS for a pNFS server.

The DS(s) must all be mounted on the MDS using the following mount options:

nfsv4,minorversion=2,soft,retrans=2

so that they can be defined as DSs in the "-p" option. Normally these mounts would be entered in the fstab(5) on the MDS. For example, if there are four DSs named nfsv4-data[0-3], the fstab(5) lines might look like:

```
nfsv4-data0:/ /data0 nfs rw,nfsv4,minorversion=2,soft,retrans=2 0 0
nfsv4-data1:/ /data1 nfs rw,nfsv4,minorversion=2,soft,retrans=2 0 0
nfsv4-data2:/ /data2 nfs rw,nfsv4,minorversion=2,soft,retrans=2 0 0
nfsv4-data3:/ /data3 nfs rw,nfsv4,minorversion=2,soft,retrans=2 0 0
```

The nfsd(8) command line option "-p" indicates that the NFS server is a pNFS MDS and specifies what DSs are to be used.
For the above fstab(5) example, the nfsd(8) nfs_server_flags line in your rc.conf(5) might look like:

```
nfs_server_flags="-u -t -n 128 -p nfsv4-data0:/data0,nfsv4-data1:/data1,nfsv4-data2:/data2,nfsv4-data3:/data3"
```

This example specifies that the data files should be distributed over the four DSs and File layouts will be issued to pNFS enabled clients. If issuing Flexible File layouts is desired for this case, setting the sysctl "vfs.nfsd.default_flexfile" non-zero in your sysctl.conf(5) file will make the **pNFSserver** do that. Alternately, this variant of "nfs_server_flags" will specify that two way mirroring is to be done, via the "-m" command line option.

```
nfs_server_flags="-u -t -n 128 -p nfsv4-data0:/data0,nfsv4-data1:/data1,nfsv4-data2:/data2,nfsv4-data3:/data3 -m 2"
```

With two way mirroring, the data file for each exported file on the MDS will be stored on two of the DSs. When mirroring is enabled, the server will always issue Flexible File layouts.

It is also possible to specify which DSs are to be used to store data files for specific exported file systems on the MDS. For example, if the MDS has exported two file systems "/export1" and "/export2" to clients, the following variant of "nfs_server_flags" will specify that data files for "/export1" will be stored on nfsv4-data0 and nfsv4-data1, whereas the data files for "/export2" will be store on nfsv4-data2 and nfsv4-data3.

```
nfs_server_flags="-u -t -n 128 -p nfsv4-data0:/data0#/export1,nfsv4-data1:/data1#/export1,nfsv4-data2:/data2#/export2,n
```

This can be used by system administrators to control where data files are stored and might be useful for control of storage use. For this case, it may be convenient to co-locate more than one of the DSs on the same FreeBSD server, using separate file systems on the DS system for storage of the respective DS's data files. If mirroring is desired for this case, the "-m" option also needs to be specified. There must be enough DSs assigned to each exported file system on the MDS to support the level of mirroring. The above example would be fine for two way mirroring, but four way mirroring would not work, since there are only two DSs assigned to each exported file system on the MDS.

The number of subdirectories in each DS is defined by the "vfs.nfs.dsdirsize" sysctl on the MDS. This value can be increased from the default of 20, but only when the nfsd(8) is not running and after the additional ds20,... subdirectories have been created on all the DSs. For a service that will store a large number of files this sysctl should be set much larger, to avoid the number of entries in a subdirectory from getting too large.

Client mounts

Once operational, NFSv4.1 or NFSv4.2 FreeBSD client mounts done with the "pnfs" option should do I/O directly on the DSs. The clients mounting the MDS must be running the nfscbd daemon for pNFS to work. Set

```
nfscbd_enable="YES"
```

in the rc.conf(5) on these clients. Non-pNFS aware clients or NFSv3 mounts will do all I/O RPCs on the MDS, which acts as a proxy for the appropriate DS(s).

Backing up a pNFS service

Since the data is separated from the metadata, the simple way to back up a pNFS service is to do so from an NFS client that has the service mounted on it. If you back up the MDS exported file system(s) on the MDS, you must do it in such a way that the "system" namespace extended attributes get backed up.

Handling of failed mirrored DSs

When a mirrored DS fails, it can be disabled one of three ways:

1 - The MDS detects a problem when trying to do proxy operations on the DS. This can take a couple of minutes after the DS failure or network partitioning occurs.

2 - A pNFS client can report an I/O error that occurred for a DS to the MDS in the arguments for a LayoutReturn operation.

3 - The system administrator can perform the pnfsdskill(8) command on the MDS to disable it. If the system administrator does a pnfsdskill(8) and it fails with ENXIO (Device not configured) that normally means the DS was already disabled via #1 or #2. Since doing this is harmless, once a system administrator knows that there is a problem with a mirrored DS, doing the command is recommended.

Once a system administrator knows that a mirrored DS has malfunctioned or has been network partitioned, they should do the following as root/su on the MDS:

```
# pnfsdskill <mounted-on-path-of-DS>
# umount -N <mounted-on-path-of-DS>
```

Note that the <mounted-on-path-of-DS> must be the exact mounted-on path string used when the DS was mounted on the MDS.

Once the mirrored DS has been disabled, the pNFS service should continue to function, but file updates will only happen on the DS(s) that have not been disabled. Assuming two way mirroring, that implies the one DS of the pair stored in the "pnfsd.dsfile" extended attribute for the file on the MDS, for files stored on the disabled DS.

The next step is to clear the IP address in the "pnfsd.dsfile" extended attribute on all files on the MDS for the failed DS. This is done so that, when the disabled DS is repaired and brought back online, the data files on this DS will not be used, since they may be out of date. The command that clears the IP address is pnfsdsfile(8) with the "-r" option.

For example:

```
# pnfsdsfile -r nfsv4-data3 yyy.c
yyy.c:	nfsv4-data2.home.rick	ds0/207508569ff983350c000000ec7c0200e4c57b2e0000000000000000	(
```

replaces nfsv4-data3 with an IPv4 address of 0.0.0.0, so that nfsv4-data3 will not get used.

Normally this will be called within a find(1) command for all regular files in the exported directory tree and must be done on the MDS. When used with find(1), you will probably also want the "-q" option so that it won’t spit out the results for every file. If the disabled/repaired DS is nfsv4-data3, the commands done on the MDS would be:

```
# cd <top-level-exported-dir>
# find . -type f -exec pnfsdsfile -q -r nfsv4-data3 {} ;
```

There is a problem with the above command if the file found by find(1) is renamed or unlinked before the pnfsdsfile(8) command is done on it. This should normally generate an error message. A simple

unlink is harmless but a link/unlink or rename might result in the file not having been processed under its new name. To check that all files have their IP addresses set to 0.0.0.0 these commands can be used (assuming the sh(1) shell):

```
# cd <top-level-exported-dir>
# find . -type f -exec pnfsdsfile {} ; | sed "/nfsv4-data3/!d"
```

Any line(s) printed require the pnfsdsfile(8) with "-r" to be done again. Once this is done, the replaced/repaired DS can be brought back online. It should have empty ds0,...,dsN directories under the top level exported directory for storage of data files just like it did when first set up. Mount it on the MDS exactly as you did before disabling it. For the nfsv4-data3 example, the command would be:

```
# mount -t nfs -o nfsv4,minorversion=2,soft,retrans=2 nfsv4-data3:/ /data3
```

Then restart the nfsd to re-enable the DS.

```
# /etc/rc.d/nfsd restart
```

Now, new files can be stored on nfsv4-data3, but files with the IP address zeroed out on the MDS will not yet use the repaired DS (nfsv4-data3). The next step is to go through the exported file tree on the MDS and, for each of the files with an IPv4 address of 0.0.0.0 in its extended attribute, copy the file data to the repaired DS and re-enable use of this mirror for it. This command for copying the file data for one MDS file is pnfsdscopymr(8) and it will also normally be used in a find(1). For the example case, the commands on the MDS would be:

```
# cd <top-level-exported-dir>
# find . -type f -exec pnfsdscopymr -r /data3 {} ;
```

When this completes, the recovery should be complete or at least nearly so. As noted above, if a link/unlink or rename occurs on a file name while the above find(1) is in progress, it may not get copied. To check for any file(s) not yet copied, the commands are:

```
# cd <top-level-exported-dir>
# find . -type f -exec pnfsdsfile {} ; | sed "/0.0.0.0/!d"
```

If this command prints out any file name(s), these files must have the pnfsdscopymr(8) command done on them to complete the recovery.

```
# pnfsdscopymr -r /data3 <file-path-reported>
```

If this command fails with the error
"pnfsdscopymr: Copymr failed for file <path>: Device not configured"
repeatedly, this may be caused by a Read/Write layout that has not been returned. The only way to get rid of such a layout is to restart the nfsd(8).

All of these commands are designed to be done while the pNFS service is running and can be re-run safely.

For a more detailed discussion of the setup and management of a pNFS service see:

https://people.freebsd.org/~rmacklem/pnfs-planb-setup.txt

SEE ALSO

nfsv4(4), pnfs(4), exports(5), fstab(5), rc.conf(5), sysctl.conf(5), nfscbd(8), nfsd(8), nfsuserd(8), pnfsdscopymr(8), pnfsdsfile(8), pnfsdskill(8)

HISTORY

The **pNFSserver** service first appeared in FreeBSD 12.0.

BUGS

Since the MDS cannot be mirrored, it is a single point of failure just as a non pNFS server is. For non-mirrored configurations, all FreeBSD systems used in the service are single points of failure.