Reexporting NFS filesystems

Overview

It is possible to reexport an NFS filesystem over NFS. However, this feature comes with a number of limitations. Before trying it, we recommend some careful research to determine whether it will work for your purposes.

A discussion of current known limitations follows.

"fsid=" required, crossmnt broken

We require the "fsid=" export option on any reexport of an NFS filesystem. You can use "uuidgen -r" to generate a unique argument.

The "crossmnt" export does not propagate "fsid=", so it will not allow traversing into further nfs filesystems; if you wish to export nfs filesystems mounted under the exported filesystem, you'll need to export them explicitly, assigning each its own unique "fsid= option.

Reboot recovery

The NFS protocol's normal reboot recovery mechanisms don't work for the case when the reexport server reboots. Clients will lose any locks they held before the reboot, and further IO will result in errors. Closing and reopening files should clear the errors.

Filehandle limits

If the original server uses an X byte filehandle for a given object, the reexport server's filehandle for the reexported object will be X+22 bytes, rounded up to the nearest multiple of four bytes.

The result must fit into the RFC-mandated filehandle size limits:

NFSv2	32 bytes
NFSv3	64 bytes
NFSv4	128 bytes

So, for example, you will only be able to reexport a filesystem over NFSv2 if the original server gives you filehandles that fit in 10 bytes--which is unlikely.

In general there's no way to know the maximum filehandle size given out by an NFS server without asking the server vendor.

But the following table gives a few examples. The first column is the typical length of the filehandle from a Linux server exporting the given filesystem, the second is the length after that nfs export is reexported by another Linux host:

	filehandle length	**after reexport**
ext4:	28 bytes	52 bytes
xfs:	32 bytes	56 bytes
btrfs:	40 bytes	64 bytes

All will therefore fit in an NFSv3 or NFSv4 filehandle after reexport, but none are reexportable over NFSv2.

Linux server filehandles are a bit more complicated than this, though; for example:

- The (non-default) "subtreecheck" export option generally requires another 4 to 8 bytes in the filehandle.
- If you export a subdirectory of a filesystem (instead of exporting the filesystem root), that also usually adds 4 to 8 bytes.
- If you export over NFSv2, knfsd usually uses a shorter filesystem identifier that saves 8 bytes.
- The root directory of an export uses a filehandle that is shorter.

As you can see, the 128-byte NFSv4 filehandle is large enough that you're unlikely to have trouble using NFSv4 to reexport any filesystem exported from a Linux server. In general, if the original server is something that also supports NFSv3, you're *probably* OK. Re-exporting over NFSv3 may be dicier, and reexporting over NFSv2 will probably never work.

For more details of Linux filehandle structure, the best reference is the source code and comments; see in particular:

- include/linux/exportfs.h:enum fid_type
- include/uapi/linux/nfsd/nfsfh.h:struct nfs_fhbase_new
- fs/nfsd/nfsfh.c:set_version_and_fsid_type
- fs/nfs/export.c:nfs_encode_fh

Open DENY bits ignored

NFS since NFSv4 supports ALLOW and DENY bits taken from Windows, which allow you, for example, to open a file in a mode which forbids other read opens or write opens. The Linux client doesn't use them, and the server's support has always been incomplete: they are enforced only against other NFS users, not against processes accessing the exported filesystem locally. A reexport server will also not pass them along to the original server, so they will not be enforced between clients of different reexport servers.