NFS Client

The NFS client

The NFS version 2 protocol was first documented in RFC1094 (March 1989). Since then two more major releases of NFS have been published, with NFSv3 being documented in RFC1813 (June 1995), and NFSv4 in RFC3530 (April 2003).

The Linux NFS client currently supports all the above published versions, and work is in progress on adding support for minor version 1 of the NFSv4 protocol.

The purpose of this document is to provide information on some of the special features of the NFS client that can be configured by system administrators.

The nfs4_unique_id parameter

NFSv4 requires clients to identify themselves to servers with a unique string. File open and lock state shared between one client and one server is associated with this identity. To support robust NFSv4 state recovery and transparent state migration, this identity string must not change across client reboots.

Without any other intervention, the Linux client uses a string that contains the local system's node name. System administrators, however, often do not take care to ensure that node names are fully qualified and do not change over the lifetime of a client system. Node names can have other administrative requirements that require particular behavior that does not work well as part of an nfs_client_id4 string.

The nfs.nfs4_unique_id boot parameter specifies a unique string that can be used instead of a system's node name when an NFS client identifies itself to a server. Thus, if the system's node name is not unique, or it changes, its nfs.nfs4_unique_id stays the same, preventing collision with other clients or loss of state during NFS reboot recovery or transparent state migration.

The nfs.nfs4_unique_id string is typically a UUID, though it can contain anything that is believed to be unique across all NFS clients. An nfs4_unique_id string should be chosen when a client system is installed, just as a system's root file system gets a fresh UUID in its label at install time.

The string should remain fixed for the lifetime of the client. It can be changed safely if care is taken that the client shuts down cleanly and all outstanding NFSv4 state has expired, to prevent loss of NFSv4 state.

This string can be stored in an NFS client's grub.conf, or it can be provided via a net boot facility such as PXE. It may also be specified as an nfs.ko module parameter. Specifying a uniquifier string is not support for NFS clients running in containers.

The DNS resolver

NFSv4 allows for one server to refer the NFS client to data that has been migrated onto another server by means of the special "fs_locations" attribute. See RFC3530 Section 6: Filesystem Migration and Replication and Implementation Guide for Referrals in NFSv4.

The fs_locations information can take the form of either an ip address and a path, or a DNS hostname and a path. The latter requires the NFS client to do a DNS lookup in order to mount the new volume, and hence the need for an upcall to allow userland to provide this service.

Assuming that the user has the 'rpc_pipefs' filesystem mounted in the usual /var/lib/nfs/rpc_pipefs, the upcall consists of the following steps:

1. The process checks the dns_resolve cache to see if it contains a valid entry. If so, it returns that entry and exits.
2. If no valid entry exists, the helper script '/sbin/nfs_cache_getent' (may be changed using the 'nfs.cache_getent' kernel boot parameter) is run, with two arguments: - the cache name, "dns_resolve" - the hostname to resolve
3. After looking up the corresponding ip address, the helper script writes the result into the rpc_pipefs pseudo-file '/var/lib/nfs/rpc_pipefs/cache/dns_resolve/channel' in the following (text) format:

   "<ip address> <hostname> <ttl>n"

   Where <ip address> is in the usual IPv4 (123.456.78.90) or IPv6 (ffee:ddcc:bbaa:9988:7766:5544:3322:1100, ffee::1100, ...) format. <hostname> is identical to the second argument of the helper script, and <ttl> is the 'time to live' of this cache entry (in units of seconds).

   > **Note**
   >
   > If <ip address> is invalid, say the string "0", then a negative entry is created, which will cause the kernel to treat the hostname as having no valid DNS translation.

A basic sample /sbin/nfs_cache_getent

```
#!/bin/bash
#
ttl=600
#
cut=/usr/bin/cut
getent=/usr/bin/getent
rpc_pipefs=/var/lib/nfs/rpc_pipefs
#
die()
{
    echo "Usage: $0 cache_name entry_name"
    exit 1
}

[ $# -lt 2 ] && die
cachename="$1"
cache_path=${rpc_pipefs}/cache/${cachename}/channel

case "${cachename}" in
    dns_resolve)
        name="$2"
        result="$(${getent} hosts ${name} | ${cut} -f1 -d\ )"
        [ -z "${result}" ] && result="0"
        ;;
    *)
        die
        ;;
esac
echo "${result} ${name} ${ttl}" >${cache_path}
```