

NFSv4 Administration

Jonathan Lyard {jonathan.lyard@bull.net}

Tony Reix {tony.reix@bull.net}

Version 0.1

April 18, 2006

Contents

1	Administration requirements	2
1.1	Configuration	2
1.2	Monitoring	3
2	Commands and files used to manage and monitor NFSv4	3
2.1	Server administration	3
2.2	Client administration	4
3	Server administration and monitoring	5
3.1	Managing exported file systems	5
3.2	Monitoring and managing NFSv4 servers	7
4	Client administration	8
4.1	Mount management	8
4.2	Monitoring and managing NFSv4 clients	10
5	Additional features	11
5.1	Security	11
5.2	Id mapping	11
5.3	Migration, Replication	11
5.4	ACLs	12

Abstract

NFSv4 (Network File System) implementation for GNU/Linux is currently being developed by CITI (University of Michigan). Bull contributes to the continuous testing activity of CITI releases and the Linux administration of this new protocol. Chapter 1 defines the administration requirements from different view points. Chapter 2 describes commands and files we will be using to manage and monitor NFSv4. Chapter 3 introduces server administration and monitoring tasks while Chapter 4 deals with similar tasks on client side. Finally, in Chapter 5, we discuss about additional features including Security, ID mapping, Migration and Replication capabilities, and ACLs.

This help us defining updates that has to be done in WebMin (a web-based interface for system administration), Nagios (enterprise-class monitoring solutions for hosts, services, and networks) and SBLIM NFSv4 provider (Standards Based Linux Instrumentation for Manageability). Any remarks or comments would be appreciated. We also hope that this draft could give the main guidelines for anyone getting involved in NFSv4 administration development.

Ω : indicates point we would mainly like comments on.

1 Administration requirements

NFSv4 is a network file system protocol aimed to be widely deployed and must nicely interoperate with different operating systems (GNU/Linux, Solaris, AIX, Windows...). Moreover, it will not be limited to LAN networks like NFSv3 since it has been designed to operate on WAN/Internet networks.

Regarding the potential number of nodes that will be using NFSv4 to share file systems, it is mandatory to provide administration tools for ones who will have to administrate and deploy configurations.

We start by stating administrators' needs so as to define features we should offer.

1.1 Configuration

First of all, NFSv4 administrators may want to build an NFSv4 network without learning every commands. They will need graphical user interfaces (GUI) to help configuring their NFSv4 server and client. WebMin is a relevant example fulfilling this requirement.

Administrators getting involved in NFS administration can use GUI to discover and learn the included features. They will use an administration tool to get familiar with the different mechanisms.

Some other administrators may prefer to configure by command line interfaces (CLI). Thus they will not necessarily need an advanced graphical interface and they will use existing NFSv4 commands. But, when managing large networks, they will certainly want to deploy or duplicate the same configuration on their nodes. In such scenarios, they will need dedicated tools to assist deployment.

Finally, it is always convenient to get a sort of "global view" interface to manage NFSv4 configurations on each node and to be able to change some properties remotely. Indeed, even expert administrators can get lost when operating on a large set of NFSv4 clients/servers.

1.2 Monitoring

When managing a large number of NFSv4 nodes and/or when availability is critical, it becomes obvious that monitoring tools are essential.

An administrator should be able to view if all NFSv4 servers are available on a network. He should also need to see if there bad errors occurred or how performance is.

In specific cases, it could also be useful to monitor NFSv4 clients. For instance, in a cluster, we want to ensure that NFSv4 client daemons are always up and that the mounted file systems does not flap.

2 Commands and files used to manage and monitor NFSv4

2.1 Server administration

2.1.1 Export a file systems

`Exportfs` is a command used to export, unexport or reexport file systems, according to either the file `/etc/exports` or to the arguments. It should be highlighted that the file is indicative and not authoritative since mountd read information in `/var/lib/nfs/etab`.

`Exportfs` command can be used to read entries from `/etc/exports`, write to `/var/lib/nfs/etab` and push them into the kernel (`exportfs -a`).

Discussion on `/etc/exports` syntax will come later as well as the set of options.

2.1.2 Unexport a file systems

`exportfs -u <directory>` unexports the given directory.

2.1.3 Exported file systems

Information about exported file systems can be found in the following files :

- ◆ `/var/lib/nfs/xtab` : lists which file systems are exported to whom (used only if `/proc` does not exist, else see `/proc/fs/nfs/exports`).
- ◆ `/proc/fs/nfs/exports` : lists which file systems are exported to whom. Data are taken from the kernels tables of exported file systems.
- ◆ `/var/lib/nfs/rmtab` : lists which file systems are actually mounted by certain clients at the moment. Entries are added to this file when mountd replies to a successful mount request, and are removed when mountd receives an unmount or unmountall request. Some entries are used by `exportfs` which instantiates wild card exports to create specific host exports to give to the kernel.

- ◆ `/var/lib/nfs/etab`: lists currently exported file trees and their options. It is in a somewhat different format than the `/etc/exports` file in that each line lists only one path and one client, and the client has all export options explicitly listed. It is the authoritative list of what should be exported to where.

Commands that can be used to extract information from the former files are :

- ◆ `exportfs -v`: lists exported file systems
- ◆ `showmount -e`: lists exported file systems (less detailed)
- ◆ `showmount -a`: lists the clients hostname and the mounted file systems

There is a problem to get which file systems are exported to whom at a given time since entries are not removed when a client unmounts (though, according to man pages, it should be!). We can just get a log of who mounted what on a server.

2.1.4 Stats and logs

`nfsstat` shows statistics about NFSv4 server (works only in latest CITI releases). A python script written by Chuck Lever (Netapp) can be used to get more stats.

Server logs can be extracted from `/var/log/messages` (`grep nfsd` and `knfsd`).

2.1.5 Daemon states

`rpcinfo` shows if the NFS daemon is running, which versions are supported (2,3,4) and which protocol is used (TCP/UDP).

`/etc/rc.d/init.d/nfs status` gives the state of `nfsd` and `mountd`.

`ps -ax | grep <daemon_name>` can be used to check if any given daemon is running.

2.2 Client administration

2.2.1 Mount a file system

`mount` is used to mount a file system (`-t nfs4` option to specify NFSv4 file system). Helpful options will be discussed in client administration. `Mount` also lists the mounted file systems.

2.2.2 Mounted file systems

`mount -t nfs4` displays all mounted file. `Showmount` cannot be used like in NFSv3 since a server exports a pseudo-file system and no more a list of directories.

- ◆ `/etc/mtab` contains the list of currently mounted file systems.
- ◆ `/proc/mounts` (if `/proc` exists) contains data similar to `/etc/mtab` (`mtab` has more information like mount options but is not necessarily up-to-date)
- ◆ `/etc/fstab` displays file systems mounted at each startup.

2.2.3 Unmount a file system

`umount` is used to unmount a file system.

2.2.4 Automount

We have to evaluate if `automount` is useful for NFSv4.

3 Server administration and monitoring

3.1 Managing exported file systems

3.1.1 Features to be provided

This is a non-exhaustive list of what should be supported in an administration tool managing NFSv4 exports :

- ◆ Configuration file name (i.e. `/etc/exports`)

And for each export :

- ◆ If the export is the virtual root (`fsid=0`)
- ◆ A permission bit (read-only / read-write)
- ◆ List of hosts allowed to access this exported file system and an ordered list of security flavors associated to this host.
 - Security flavors list will be used during NFSv4 security negotiation. Flavors should be taken from :

- sys
- krb5 (authentication, integrity and privacy modes)
- spkm (3 modes)
- lipkey (3 modes).
- o Hosts can be specified by either :
 - everyone (nothing or *) : */export/dir (...)*
 - hostname : */export/dir host1.foo.net(...)*
 - IPv4 network : */export/dir 129.138.0.1/255.255.255.0(...)*
 - IPv6 network : */export/dir fe80::204:76ff:fe19:76b7/64(...)*
 - NIS net group (Is it still needed ?)
- ◆ Extract information about binds to provide the original path of the exported directory (to export a directory which is not under the NFSv4 root we use mount -bind)

We should handle several options (we still have to clearly define which ones are truly used by NFSv4 Ω) :

- ◆ Subtree checking (*subtree_check / no_subtree_check*)
 - o Hide the file system (*hide / no_hide*)
 - o Writes synchronization :
 - Immediately synchronize all writes (*sync / nosync*). Default has changed to sync since last version but it has to be explicitly specified otherwise exportfs raise a warning.
 - No delay in committing a write request to disc (*no_wdelay / wdelay*)
 - o Rights :
 - Client must be on secure port (*secure/insecure*)
 - Trust every remote user (*insecure, no_root_squash*)
 - Trust every remote user except root (*insecure, root_squash*)
 - Do not trust any remote user (*insecure, all_squash*)
 - Change default value of anonymous account (*anonuid=value*)
 - Change default value of anonymous group (*anongid=value*)

3.1.2 Exports Syntax

A specific */etc/exports* syntax has been defined under AIX and Solaris operating systems. For the time being, the GNU/Linux *exports* syntax is still subject to changes. Since several administration tools are based on the */etc/exports* syntax, it is crucial to write a formal description. Of course, we can contribute to solve this issue : we can write an EBNF syntax (for instance) in agreement with the community. Ω

To illustrate the problem, we have seen that WebMin uses a syntax similar to NFSv3's but SBLIM NFSv4 provider uses a slightly different one. Once "normalized", we will have to adapt those tools to deal with the new syntax.

3.2 Monitoring and managing NFSv4 servers

3.2.1 Daemons

To use NFSv4 on a server, the following daemons have to be launched :

- ◆ `rpc.nfsd`
- ◆ `rpc.mountd` (temporary ?Ω)
- ◆ `rpc.idmapd` for ID mapping
- ◆ `rpc.svcgssd` if using security (optional but recommended)

A monitoring tool should display daemons state and eventually raise alarms if they go down.

Tools should provide a way to start, stop, restart services and an interface for basic daemon options. There is a proposal :

- ◆ `rpc.nfsd`
 - Specify a different port to listen to NFS requests (default is 2049) (`-p`)
 - Specify the number of NFS server threads (default is a single thread, recommended is 8)
- ◆ `rpc.idmapd`
 - Server-only: perform no id mapping for any NFS client, even if one is detected (`-S`)

Other options can be specified via `/etc/idmapd.conf`. We will see later how this configuration file can be managed.

- ◆ `rpc.svcgssd`
 - Set pipefs directory (`-p`)

Management of this set of daemons can be inspired from `/etc/rc.d/init.d/nfs` which is the script managing start, restart, stop, status (...) of NFS server daemons.

A facility can be used to monitor daemons remotely thanks to `rpcinfo -p <host>` in order to display RPC services run on a remote machine. A broadcast options also exists to query machines on the LAN (broadcast domain) running a given RPC service version.

3.2.2 Errors

We have identified some errors to be handled in a monitoring tool :

- ◆ RPC errors (the bad* strings read from the `nfsstat` command)
- ◆ Problems found in the logs (extract from `/var/log/messages`)

3.2.3 Performances

- ◆ CPU and memory utilization of `nfsd` daemon (and possibly other daemons)
- ◆ Number of effective `nfsd` threads (read from `/proc/net/rpc/nfsd`)

3.2.4 Statistics

- ◆ Number of calls per NFSv4 procedure (read from `nfsstat` command + patch)
 - ◆ And possibly all other information displayed by `nfsstat` (as soon as they are supported in NFSv4)
-

4 Client administration

4.1 Mount management

Providing a user interface to manage mounts can help adopt NFSv4. There are several ways :

- ◆ Integrate `mount` interface in GNU/Linux distribution (Fedora, Suse, Debian, Mandriva, Gentoo...). It can be a specific distribution application or a KDE/Gnome one. KDE control center integrate a module for “File sharing” managing Samba and NFS(v.2,3) mounts.
- ◆ Write a specific application to manage NFSv4 mounts. But this may be too difficult to make the end-user adopt it. They will certainly prefer to have a module in their common administration tool since they generally do not only manage NFS.
- ◆ Add a dedicated module to classical administration tools. Bull had already provided a mount module for WebMin (a classical web based management tool). We will update this module to current NFSv4 implementation.

4.1.1 Mounting NFSv4 file systems

Client administration tools need to enable users to mount and unmount file systems in a more friendly way than command line interfaces.

To mount a remote file system, user should choose between :

- ◆ Mounting the file system immediately
- ◆ Mounting the file system at system restart
- ◆ Mounting the file system immediately and at subsequent system restart

We also have to query the user for some required arguments :

- ◆ NFS server address by either IPv4 address, netname, DNS name, IPv6 address (? Ω)
- ◆ NFS directory (in general '/' to mount the pseudo file system)
- ◆ Local mount point : local directory where the file system should be mounted
- ◆ Authentication mode (security flavors as described in 3.1.1)

We also need to support common mount options :

- ◆ Read-only or read-write (*ro/rw*)
- ◆ Synchronous or asynchronous file system I/O (*sync/async*).
- ◆ Allow device files (*dev/nodev*)
- ◆ Allow execution of binaries (*exec/noexec*)
- ◆ Allow usage of setuid bit for program (*suid/nosuid*)
- ◆ Allow an ordinary user to mount the file system (*user/nouser*)
- ◆ No automount : `mount -a` will not mount the file system (*noauto*)

And some NFS specific mount options :

- ◆ Specify NFSv4 port if different from 2049 (*port*)
- ◆ Retry mount in background if the first first attempt fail due to timeout (*bg/fg*)
- ◆ Return error to program when an operation has a major timeout (*soft/hard*)
- ◆ Set timeout before first retransmission (then, double up to 60s) (*timeo* (default=0.7s))
- ◆ Set number of retransmission (trigger major timeout) (*restrans* (default=3))
- ◆ Allow user to interrupt (via signal) after major timeout (*intr/nointr*)
- ◆ Set read and write buffers size Supported Rsize, wsize (recommanded=32768)
- ◆ Cache timer (for regular file and directories) (*acregmin, acregmax, acdirmin, acdirmax*)
- ◆ Number of minutes to retry a mount before giving up (*retry* (default=1 week!))
- ◆ Suppress the retrieval of attributes when creating a file (*nocto*)
- ◆ Disable all forms of attributes caching (*noac*)
- ◆ Set a specific callback address (multi-homed client) (*clientaddr*) (currently ignored ?)

4.1.2 Unmounting NFSv4 file system

To unmount a file system, we just have to query the user for the local mount point (*umount* command).

4.1.3 Mount visualization

Mount visualization aims at providing the end-user a clear view of his mounts and their associated options. Information can be extracted from `showmount`, `mount` or `/etc/fstab` (for `mount` at start up).

Our first idea was to provide a “global view” tool displaying what each client in our network mounts. Unfortunately, it does not seem feasible since that entails having a provider on each machine extracting information and answering queries sent by the monitoring server. On clusters, we could install an agent on each node but what if we manage a WAN/Internet NFSv4 network ?

Getting client’s mounts from NFSv4 servers could solve this problem, but it seems difficult to keep track of clients (we only have an opaque client ID) and to extract information from the kernel. In user land, we can still use `showmount` on the server which assumes identifying a client by its IP address (problematic with NAT, multi-homed environments...) ?

Consequently, the best would be to manage mounts with a tool installed on the client itself (via web interface this can still be administrated remotely). This tool could be integrated in distros to manage NFSv4 mounts like NFSv2-3 as well as Samba ones (see KDE Control Center).

We will enhance the WebMin NFSv4 module which provides a Web interface used to view and modify mounts.

4.2 Monitoring and managing NFSv4 clients

Similarly to servers, we can monitor NFSv4 clients. But, as discussed earlier, this requires to have an agent on each client.

In classical NFSv4 application the client will see the problem as soon as he is impacted. Needs for client monitoring will thus certainly deal mainly in assisting the user at solving the problem.

We emphasize that client going down is in general much less critical than servers issues. Possibly the user simply shuts down the machine or voluntary unmounts the server root. For some very specific applications (i.e. cluster), it can be necessary to check that all clients are up and mounting a file system but it turns out difficult to design a general monitoring application for NFSv4 clients.

5 Additional features

5.1 Security

Although using security is not mandatory, it is a major advantage when adopting NFSv4. For the time being, only Kerberos is really in working order. Support for SPKM-3 and LIPKEY should come later.

The main difficulty when setting up a Kerberos-NFSv4 environment is to configure Kerberos itself. Some administration tools already integrate a GUI to help configuring this security architecture.

Regarding NFSv4 administration, there is nothing special to add (we have already defined how to support Kerberos in NFS exports and mounts).

We will evaluate later if special administration for SPKM-3 and LIPKEY should be necessary.

5.2 Id mapping

There is currently no support for id mapping in SBLIM NFSv4 provider. NFSv4 supports two methods for id mapping : `nsswitch` (Name Service Switch) and `umich_ldap`. As the latter is experimental, it is too early to provide administration tools the syntax may still evolve.

However, we would like to provide administration for the stable `nsswitch` method.

We can provide an interface to file `/etc/idmapd.conf` with :

- ◆ choice of translation method (here `nsswitch`)
- ◆ location of RPC pipefs
- ◆ domain name
- ◆ manual mapping including `nobody-user` and `nobody-group`

We also have to decide if an interface for `/etc/nsswitch.conf` is needed. (Is it well supported in current NFSv4 implementation ?)

Once `umich_ldap` validated and stable, we can support it in NFSv4 provider with :

- ◆ LDAP server information : addresses, base, name attribute, group attribute, principal attribute

5.3 Migration, Replication

Regarding the current state of those features in NFSv4 GNU/Linux implementation, it is a bit early to provide administration tools supporting them.

AIX implementation comes with several commands to administrate migration and replication features. Something similar may be done in GNU/Linux.

5.4 ACLs

NFSv4 ACLs have been validated and we can support them. Basically, this will certainly consist in accessing and modifying POSIX ACLs that are mapped onto NFSv4 ones.

This is feasible. But probably it should not be managed within NFSv4 modules. For instance, WebMin manages POSIX ACLs outside NFS modules since the use is not restricted to NFSv4.

However, it could be interesting to see which exports are actually supporting ACL - i.e. if the file-system has been mounted with ACL support (ACL option in /etc/fstab).