IBM

# Using the Linux NFS Client with IBM System Storage N series

- Kernel releases and mount options
- Tuning the Linux client for performance and reliability
- Utilities to support advanced NFS features



Alex Osuna
Eva Ho
Chuck Lever

Redbooks

ibm.com/redbooks

International Technical Support Organization

# Using the Linux NFS Client with IBM System Storage N series

May 2007



SG24-7462-00

**Note:** Before using this information and the product it supports, read the information in “Notices” on page v.



**First Edition (May 2007)**

This edition applies to Version 7.1 of Data ONTAP and later.

# Contents

# Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
*IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.*

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:** INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

# Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

| | |
|---|---|
| Redbooks (logo)  ® | System Storage™ |
| IBM® | Tivoli® |

The following terms are trademarks of other companies:

Oracle, JD Edwards, PeopleSoft, Siebel, and TopLink are registered trademarks of Oracle Corporation and/or its affiliates.

Network Appliance, Data ONTAP, and the Network Appliance logo are trademarks or registered trademarks of Network Appliance, Inc. in the U.S. and other countries.

Solaris, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Active Directory, Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, Itanium, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

# Preface

This book will help you get the best from your Linux® NFS clients when used in an environment that includes IBM® System Storage™ N series products. It begins with a general discussion of Linux distributions and kernels, and includes a comparison of some of the features of the 2.4 and 2.6 kernels. A detailed discussion of mount options and network protocols is presented to guide you through configuration tasks. Procedures for tuning your Linux clients and steps to diagnose performance and reliability problems are presented, along with information about additional resources you can consult if necessary. Finally, steps for configuring and using utilities that support advanced NSF features are provided.

This document is appropriate for customers, systems engineers, technical marketing engineers, and customer support personnel who install and configure Linux systems that use NFS to access N series systems and network caches.

## The team that wrote this book

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, Tucson Center.

**Alex Osuna** is a project leader at the International Technical Support Organization, Tucson center. He writes extensively on storage and has taught worldwide on all areas of storage. Before joining the ITSO 2 years ago he was a Systems Engineer with Tivoli®. Prior to that he held positions with ATS, Customized Operational Services, Special Bids, Service Planning and Field Engineering. He holds over 10 certifications with IBM, Microsoft® and Redhat.

**Eva Ho** is an IT Specialist with IBM Systems Technology Group. She has over 22 years of experience working with servers, networking products, IBM Network Attached Storage appliances, and IBM System Storage N series. She is the technical team lead for the IBM WW N series PFE support team in Research Triangle Park, North Carolina. Eva has system storage certification with IBM. She was a participant in developing IBM Storage Networking Solutions V1 and V2 Certification test. Eva holds a master's degree in Computer Science.

**Chuck Lever** worked for Network Appliance™ Inc. at the time he contributed to this document.

## Become a published author

Join us for a two- to six-week residency program! Help write a book dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll have the opportunity to team with IBM technical professionals, Business Partners, and Clients.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

`ibm.com/redbooks/residencies.html`

# Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this or other books in one of the following ways:

- Use the online **Contact us** review form found at:

  ibm.com/redbooks

- Send your comments in an e-mail to:

  redbooks@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

1

# Linux NSF client overview and solution design considerations

This chapter provides an introduction to the Linux NSF client and assists you in optimizing it for implementation. It also gives tips on mount options for various clients, and describes the configuration issues to consider when selecting a network transport protocol and specifying the size of read and write operations.

## 1.1 Introduction

More and more N series customers recognize the value of Linux in their enterprises. Historically, the Linux NFS client has trailed the rest of Linux in providing the level of stability, performance, and scalability that is appropriate for enterprise workloads. In recent times, however, the NFS client has improved considerably and continues to improve in performance and ability to work even under degraded network conditions.

This document addresses several areas that concern those who are planning a new Linux deployment or are administering an existing environment that contains Linux NFS clients accessing IBM System Storage N series products. These areas include:

- What level of performance and stability to expect from Linux NFS clients
- How to tune Linux NFS clients to perform well with N series products
- How to diagnose client and network problems that involve Linux NFS clients
- Which network interfaces and drivers work best in customer's existing environment
- How to configure other services required to provide advanced NFS features
- Where to find more tuning and diagnostic information on N series Web site and the Internet

## 1.2 Deciding which Linux NFS client is right for you

In this section we describe some of the things you should consider when designing a solution using a Linux NSF client. Except for clients running Oracle® databases[1], specific Linux kernel releases or distributions are not recommended at this time. Instead of recommending one or two releases that work well, we provide some guidelines to help you decide among the many Linux distributions and releases, and we provide advice on how to make your Linux NFS clients work their best.

The reasons that we do not recommend specific Linux releases are the following:

- There are too many distributions and releases to qualify them all. There are more than a half-dozen distributions and thousands of different kernel releases. Added to that are the many different versions of user-level helper applications (such as the Linux mount command). Because all of these are open source, you can modify or replace any part of your client (Figure 1-1).

[1] You can view N series compatibility with Oracle at the following Web site: http://www-03.ibm.com/servers/storage/oraclecompatibility.html

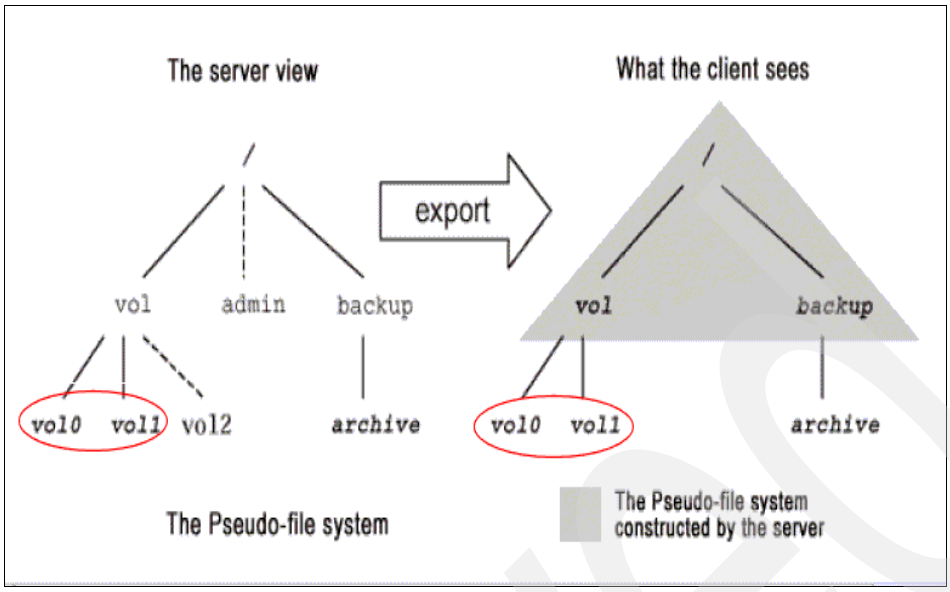


*Figure 1-1 An example NFS client mounting*

- Many hardware and application vendors specify a small set of releases or a single release and distribution that is certified and supported. It would be confusing for us to recommend one particular kernel or distribution when your hardware vendor recommends another, and your application vendor certifies yet a third.
- Some applications are more sensitive to NFS client behavior than others. Recommending a particular Linux NFS client depends on the applications you want to run and what your performance and reliability requirements are.

Before we begin to focus on technical details, we cover some basic technical support challenges specific to Linux. The Linux NFS client is part of the Linux kernel. Because Linux is open source, you might think that it is easy to provide Linux kernel patches to upgrade the NFS client. In fact, providing a patch that fixes a problem or provides a new feature can be complicated by several facts of life in the Linux and open source worlds.

There are many different parts to Linux, but the two we are concerned about are the *distribution* and the *kernel*:

- The distribution is the set of base operating system files that are included when customers install a Red Hat or SUSE Linux distribution on their hardware. This includes commands, applications, and configuration files.
- A kernel comes with a distribution, but customers can replace it, usually without affecting other files provided in the distribution.

## 1.2.1 Identifying kernel releases

The version number of a Linux distribution and the release number of a Linux kernel use different naming schemes. While planning a distribution, each distributor chooses a particular kernel release (for example, 2.6.5) and adds some modifications of its own before placing the kernel into a distribution. To reduce the amount of variation they encounter in their support contracts, distributors support only a small set of kernels, most of which are carefully designed for a specific distribution.

Because the NFS client is part of the kernel, updates to the NFS client require that you replace the kernel. Technically, it is easy to replace a kernel after a distribution is installed, but Linux customers risk losing distributor support for their installation if they install a kernel that was not built by the distributor. For this reason, we do not recommend specific Linux patches or kernel versions. Often support contracts constrain customers so they cannot install a particular patch until their chosen distributor provides a supported kernel that includes the patch.

The current kernel is released in two branches, known as the “stable” branch and the “development” branch. The stable branch is ostensibly the branch that is hardened, reliable, and has unchanging program interfaces, while the development branch can be (and often is) unstable and sometimes is even unbootable. Stable branches have even minor release numbers, such as 2.4, while development branches have odd minor release numbers, such as 2.5.

The previous stable branch is 2.4, the latest stable branch is 2.6, and the latest development branch has not been opened yet. Linux kernels are not published on a time-based schedule. Kernel revisions are released when the branch maintainer decides they are ready. New features and API changes are allowed in development kernels, but there is no schedule for when such a kernel will become a stable release. Development branches have historically taken two years to 30 months to become stable branches.

It is for this reason that there is often significant pressure to add new features to stable releases instead of working them into development releases. To expedite the addition of new features, kernel developers have recently changed the way stable and development branches are treated. As of this writing, the 2.6 kernel is now being used as a development branch while distributors are treating their own kernels as the stable branches.

### 1.2.2 Current Linux distributions

As mentioned previously, distributions are numbered differently than kernels. Each distributor chooses its own numbering scheme. When describing your Linux environment to anyone, be sure to list both the distribution release number and the kernel release number. Distributors usually append another number on the end of their kernel versions to indicate which revision of that kernel is in use. For instance, Red Hat shipped kernel 2.4.18-3 with its 7.3 distribution, but made several other errata kernels available over time to fix problems in its kernel: 2.4.18-10, 2.4.18-17, 2.4.18-27, and so on.

An important trend in commercial Linux distributions is the existence of enterprise Linux distributions. Enterprise distributions are quality-assured releases that come with special support contracts. Not all customers need this level of support, however. Red Hat recently changed its product line, dropping the professional series of distributions in favor of an openly maintained distribution called *Fedora*. Fedora is intended for developers and customers who can tolerate some instability on their Linux systems. SUSE continues to sell an enterprise distribution as well as a less expensive desktop product line. Distributions such as Mandrake and Debian are ideal for customers looking for a low-cost general purpose Linux distribution but who have enough expertise to support themselves.

You are strongly advised to always use the latest actively maintained distributions available. Customers running older unsupported distributions no longer get the benefits of security fixes and quick bug fixes on their Linux systems. Most Linux distributors will not address bugs in older distributions at all. Especially if your clients and storage systems are not protected by a firewall, it is important for you to stay current with the latest Linux errata patches available for your distribution.

To find out which kernel your clients run, use the **`uname -r`** command (Figure 1-2).

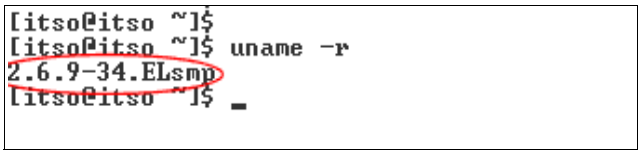


*Figure 1-2 Verifying the kernel version of your Linux client*

Linux kernels built from community source code usually have only three or four dot-separated numbers, like 2.6.8.1. Distributors generally add a hyphen and more version numbers (in this case, -34), which indicate that additional patches over the community source base have been applied. The keyword on the end, such as “hugemem” or “smp,” shows additional hardware capabilities for which this kernel was built.

## 1.2.3 The NFS client in the 2.4 kernel

The NFS client in release 2.4 kernel has many improvements over the older release 2.2 client, most of which address performance and stability problems. The NFS client in kernel releases later than 2.4.16 include significant changes for performance and stability improvements.

### Linux memory management

In Linux, physical memory is mapped by the kernel into the virtual address space. The kernel cannot access memory that has not been mapped into its address space. The memory management unit (MMU) of the processor converts the virtual address (the address referenced within a process in user-space) to the proper physical address based on the page tables created by the kernel.

The Linux kernel splits the 4 GB virtual address space of a process into two parts: 3 GB (the user-space virtual addresses) and 1 GB (reserved for the kernel virtual addresses). All available kernel virtual space addresses are mapped to the available physical memory by the kernel, which means 1 GB of virtual addresses can translate into a maximum of 1 GB of physical memory. However, some area of physical memory is reserved for storing kernel data structures, such as memory map and page tables. On an x86 machine, 128 MB of memory is reserved (out of the 1 GB physical memory) and only the kernel can access it. The kernel virtual address in this 128 MB is not mapped to physical memory. This leaves a maximum of 896 MB for user-space virtual addresses mapping. So, even if one has 1 GB of physical RAM, only 896 MB will actually be available.

If more than 1 GB of physical RAM is used, the high (additional) memory has to be directly mapped into the virtual address space using a special kernel compile option, CONFIG_HIGHMEM, before it can be accessed. The following options are available:

- CONFIG_HIGHMEM solution for using up to 4 GB of memory by using the following:
  - Use the kmap() function to create a permanent mapping. This function may sleep, so it should not be used in interrupt context.

    **Note:** The number of permanent mappings is limited. Make sure the pages mapped this way are unmapped using kunmap() when no longer needed.

  - Use the kmap_atomic() function to create temporary mappings. This function does not block, so it can be used in interrupt context.

**Note:** A temporary mapping is only available as long as the next temporary mapping. Make sure a page mapped via kmap_atomic() is unmapped using kunmap_atomic() when no longer needed.

- CONFIG_HIGHMEM solution for using up to 64 GB of memory (36 bits of address bus):

  This is enabled via the Physical Address Extension (PAE) of the PentiumPro processors. Since the virtual address space is limited to 32 bits wide, each process cannot grow beyond 4 GB. PAE addresses the 4 GB physical memory limitation and allows processors to access physical memory up to 64 GB.

The Linux NFS client has a known problem in these configurations where an application or the entire client system can hang unexpectedly when the system is experiencing heavy read- or write-intensive workloads on multiple mount points.

This issue has been addressed in the Linux community's 2.4.20 kernel release, but still persists in kernels contained in earlier distribution releases from Red Hat and SUSE. It is recommended that you either upgrade to a kernel later than 2.4.20, or upgrade to the most recent 2.4-based distribution you can use. For kernels earlier than 2.4.20, there is no known workaround except to disable the CONFIG_HIGHMEM option.

### Other known issues

- Early releases of the Red Hat 7.3 kernel demonstrated a bug in the Linux IP implementation which caused IP packet reassembly errors on the storage system. It had great impact on NFS client performance when mounting over UDP since it can cause a storage system to become temporarily inaccessible.

  **Note:** Refer to http://bugzilla.redhat.com for more details on RedHat bugs 64921, 64984, and 65069. The recent errata kernels that include performance improvement fixes to address the issue are available from Red Hat.

- Earlier Linux kernels release 2.4 series demonstrated various NFS-related issues when mounting NFS over TCP. The latest 2.4-based distributions (SUSE SLES 8 SP3, Fedora Core 1, and RHEL 3.0) use kernels that have a robust NFS over TCP implementation. Kernels older than 2.4.20 can suffer from problems with NFS mounted over TCP that result from lossy networks and overloaded NFS servers. The issue can cause NFS mount points to become unusable until the client is rebooted.

  **Note:** No matter which 2.4 kernel your distribution uses, you should always start with NFS over TCP first because TCP has a number of important benefits over UDP.

## 1.2.4 The NFS client in the 2.6 kernel

During 2004, distributions based on the 2.6 kernel became available and stable enough to be deployed in production environments. SUSE SLES 9 was the first enterprise Linux distribution to use the 2.6 kernel. Red Hat has also introduced its own 2.6-based enterprise version, Red Hat Enterprise Linux 4.

### NFSv4 support

Linux 2.6 kernel includes support for the latest version of the Network Filesystem version 4 (NFSv4) protocol. Developers are still in the process of retrofitting the Linux NFS client and server implementations for this new NFS protocol version. Currently, NFSv4 supports

filesystem replication and migration on the server side and also has the ability to have the server delegate certain responsibilities to the client in caching situations, which is necessary for providing true data integrity.

Support for NFSv4 is also available with Fedora Core and RHEL 4.

### Kerberos 5 support

The 2.6 kernel also brings support for advanced authentication mechanisms such as Kerberos 5. Support for Kerberos works with NFS versions 2 and 3 as well as for NFS version 4. Kerberos authentication increases security by reducing the likelihood that user identities in NFS requests can be forged. It also provides optional facilities to ensure the integrity or privacy of communication between an NFS client and server. The Linux implementation of NFS with Kerberos is based on Internet standards; it is tested with the leading proprietary Kerberos implementations regularly. Linux NFS with Kerberos will interoperate seamlessly with N series storage systems once all NFSv4 features are uniformly implemented.

### Upgrade and deployment

The NFS client in the 2.6 kernel has demonstrated superior performance and stability over Linux NFS clients in older kernels, but as usual, customers should test before moving their production workloads onto this release of the Linux kernel.

**Note:** You should use the latest distribution and kernel available from your distributor when installing a new deployment, and attempt to keep your existing Linux clients running the latest updates from your distributor. Always check with your hardware and application vendors to be certain they support the distribution and kernel you choose to run.

## 1.3 Mount options for Linux NFS clients

This section describes the mount options available for Linux NSF clients.

To view the NFS mount options that the Linux client attempts to set on Linux, look in the /etc/fstab file (Figure 1-7 on page 11). It contains information about which filesystem to mount, where, and with what options. To understand those terms used in fstab, issue the **`man nfs`** command at a shell prompt to display the manual page, and press Enter to scroll to the next page (Figure 1-3 on page 8).

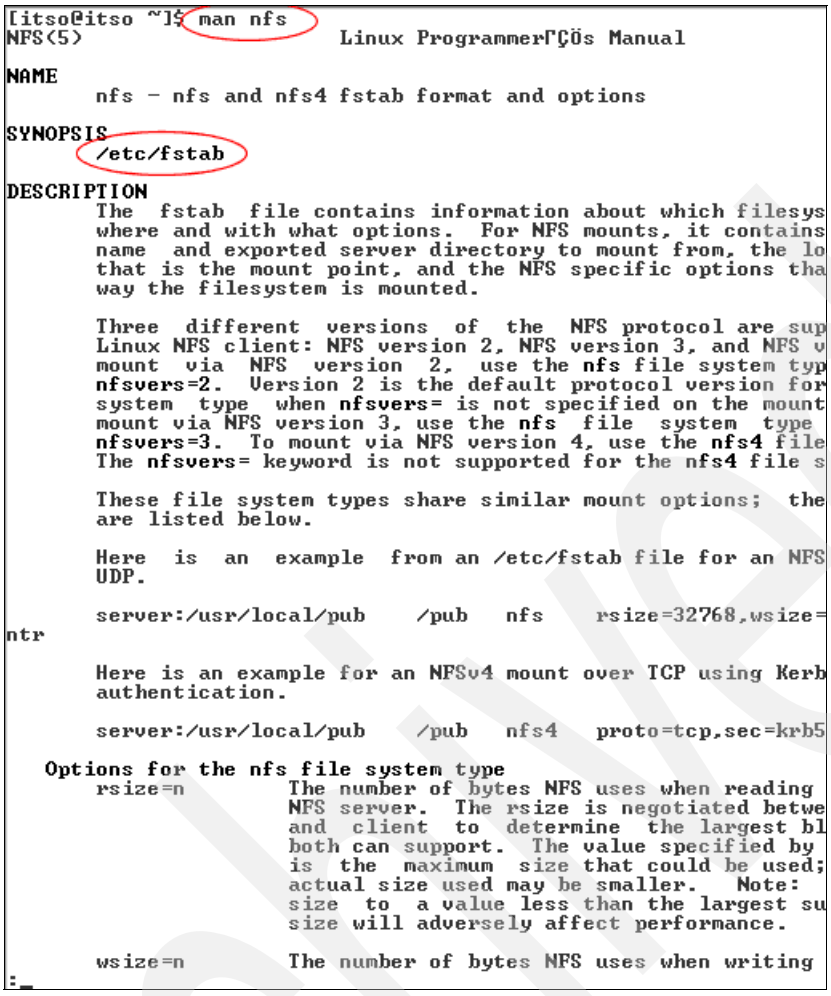


*Figure 1-3 Example of man nfs command*

Check your automounter configuration to see what defaults it uses when mounting. Running the **mount** command at a shell prompt tells you what options are actually in effect (Figure 1-4 on page 9). Clients negotiate some options, for example, the rsize and wsize options, with servers. Look in the client's /proc/mounts file to determine exactly what mount options are in effect for an existing mount.

```
[itso@itso ~]$
[itso@itso ~]$ mount
/dev/mapper/VolGroup00-LogVol00 on / type ext3 (rw)
none on /proc type proc (rw)
none on /sys type sysfs (rw)
none on /dev/pts type devpts (rw,gid=5,mode=620)
usbfs on /proc/bus/usb type usbfs (rw)
/dev/sdb1 on /boot type ext3 (rw)
none on /dev/shm type tmpfs (rw)
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
sunrpc on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw)
[itso@itso ~]$
[itso@itso ~]$ cat /proc/mounts
rootfs / rootfs rw 0 0
/proc /proc proc rw,nodiratime 0 0
none /dev tmpfs rw 0 0
/dev/root / ext3 rw 0 0
none /dev tmpfs rw 0 0
/proc /proc proc rw,nodiratime 0 0
/proc/bus/usb /proc/bus/usb usbfs rw 0 0
/sys /sys sysfs rw 0 0
none /dev/pts devpts rw 0 0
/dev/sdb1 /boot ext3 rw 0 0
none /dev/shm tmpfs rw 0 0
none /proc/sys/fs/binfmt_misc binfmt_misc rw 0 0
sunrpc /var/lib/nfs/rpc_pipefs rpc_pipefs rw 0 0
[itso@itso ~]$ _
```

*Figure 1-4 Example of mount command and /proc/mounts file*

The default NFS protocol version (2, 3, or 4) used when mounting an NFS server can change depending on what protocols the server exports, which version of the Linux kernel is running on the client, and what version of the mount utilities package your client uses. NFSv2 is the default on earlier versions of 2.4 kernels, but most modern distributions make NFSv3 the default (Figure 1-5). Make sure both the Linux client and the N series storage system use the NFSv3 protocol before trying to perform NFS mount from the Linux client. To verify the NFSv3 setting on your N series storage system, use the **options** command (Figure 1-5).

**Note:** NFSv3 support on IBM System Storage N series products is enabled by default.

```
itsotuc1>
itsotuc1> options nfs
nfs.access                      *
nfs.assist.queue.limit          40
nfs.export.allow_provisional_access on
nfs.export.auto-update          on
nfs.export.harvest.timeout      1800
nfs.export.neg.timeout          3600
nfs.export.pos.timeout          36000
nfs.export.resolve.timeout      15
nfs.hide_snapshot               off
nfs.ifc.xmt.high                16
nfs.ifc.xmt.low                 8
nfs.kerberos.enable             off
nfs.kerberos.file_keytab.enable off
nfs.locking.check_domain        on
nfs.mount_rootonly              off
nfs.mountd.trace                off
nfs.netgroup.strict             off
nfs.notify.carryover            on
nfs.per_client_stats.enable     off
nfs.require_valid_mapped_uid    off
nfs.response.trace              off
nfs.response.trigger            60
nfs.rpcsec.ctx.high             0
nfs.rpcsec.ctx.idle             360
nfs.tcp.enable                  on
nfs.udp.enable                  on
nfs.udp.xfersize                32768
nfs.v2.df_2gb_lim               off
nfs.v3.enable                   on
nfs.v4.acl.enable               off
nfs.v4.enable                   off
nfs.v4.id.domain
nfs.v4.read_delegation          off
nfs.v4.write_delegation         off
nfs.webnfs.enable               off
nfs.webnfs.rootdir              XXX
nfs.webnfs.rootdir.set          off
itsotuc1> _
```

*Figure 1-5 Default option setting for nfs.v3.enable on N series storage system*

Disable this option if there is a problem with your client when using NFSv3, and that client cannot be configured to use NFS version 3 (Figure 1-6). Valid values for this option are `on` (enabled) or `off` (disabled). To disable nfs.v3.enable, issue the following command:

```
options nfs.v3.enable off
```

```
itsotuc1>
itsotuc1> options nfs.v3.enable
nfs.v3.enable                   on
itsotuc1>
itsotuc1> options nfs.v3.enable off
itsotuc1>
itsotuc1> options nfs.v3.enable
nfs.v3.enable                   off
itsotuc1>
itsotuc1>
```

*Figure 1-6 Disable nfs.v3.enable on the N series storage system*

The `hard` mount option is the default on Linux and is mandatory if you want data integrity (Figure 1-7). Using the `soft` option reduces the likelihood of client instability during server and network outages, but it exposes your applications to silent data corruption, even if you mount filesystems read-only. If a soft timeout interrupts a read operation, the client's cached copy of the file is probably corrupted. To purge a corrupted file, it requires that the application locks and unlocks the file, that the whole filesystem is unmounted and remounted, or that another client modifies the file size or `mtime` (UNIX® jargon for "the time at which file name was last modified"). If a soft timeout interrupts a write operation, there is no guarantee that the file on the server is correct, nor is there any guarantee that the client's cached version of the file matches what is on the server.

```
[itso@itso ~]$
[itso@itso ~]$ cat /etc/fstab
# This file is edited by fstab-sync - see 'man fstab-sync' for details
/dev/VolGroup00/LogVol00 /                       ext3    defaults
 1 1
LABEL=/boot             /boot                   ext3    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
/dev/VolGroup00/LogVol01 swap                    swap    defaults
 0 0
//192.168.3.249/temp  /share  smbfs  user,password          uid=500,gid=5
00,username=Administrator  0  0
192.168.3.254:/vol/oradata  /mnt/oradata  nfs  user,dev,rsize=32768,time
o=600,tcp,suid,wsize=32768,exec,bg,noauto,hard  0  0

192.168.3.254:/vol/oralog /mnt/oralog nfs user,dev,rsize=32768,timeo=600
 tcp,suid,wsize=32768,exec,bg,noauto,hard 0  0
192.168.3.252:/etc  /filer  nfs  user,suid,dev,exec,noauto  0  0
/dev/hda                /media/cdrom            auto    pamconsole,exec,
noauto,managed 0 0
/dev/fd0                /media/floppy           auto    pamconsole,exec,
noauto,managed 0 0
```

*Figure 1-7 An example of /etc/fstab file*

A Linux client can indicate that a soft timeout has occurred in various ways. Usually system calls return EIO (I/O Error) when such a timeout occurs. You may also see messages in the kernel log suggesting that the client had trouble maintaining contact with the server and has given up. If you see a message that says the client is still trying, then the hard mount option is in effect.

As an alternative to soft mounting, consider using the `intr` option, which allows users and applications to generate signals to interrupt the NFS client file operation when it gets stuck waiting for server or network recovery. On Linux, interrupting applications or **`mount`** commands(Figure 1-8) do not always work, so sometimes rebooting your client is necessary to recover a mount point that has become stuck because the server is not available.

When running applications such as databases that depend on end-to-end data integrity, you should use `hard, nointr`. Oracle has verified that using `intr` instead of `nointr` can expose your database to the risk of corruption when a database instance is signaled (for example, during a `shutdown abort` sequence).

The soft option is useful only in a small number of cases. If you expect significant server or network instability, try using the soft option with TCP to help reduce the impact of temporary problems. When using the soft option, especially with UDP, specify a relatively large number of retries. This reduces the likelihood that very brief outages or a few dropped packets will cause an application failure or data corruption.

```
[root@itso mnt]# mount -t nfs 192.168.3.254:/vol/oradata1 /mnt/oradata
1
[root@itso mnt]# mount
/dev/mapper/VolGroup00-LogVol00 on / type ext3 (rw)
none on /proc type proc (rw)
none on /sys type sysfs (rw)
none on /dev/pts type devpts (rw,gid=5,mode=620)
usbfs on /proc/bus/usb type usbfs (rw)
/dev/sdb1 on /boot type ext3 (rw)
none on /dev/shm type tmpfs (rw)
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
sunrpc on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw)
//192.168.3.249/temp on /share type smbfs (0)
192.168.3.254:/vol/oradata1 on /mnt/oradata1 type nfs (rw,addr=192.168.3
.254)
```

*Figure 1-8 An example of NFS mount and verify the mount point*

## 1.4 Choosing a network transport protocol

Although UDP is a simple transport protocol that has less CPU and network overhead than TCP, NFS over UDP has deficiencies that are exposed on congested networks, such as routed multispeed networks, DSL links, and slow WANs, and should never be used in those environments. TCP is almost always a safe bet, especially on versions of Linux more recent than 2.4.19.

NFSv2 uses the User Datagram Protocol (UDP) to provide a stateless network connection between the client and server. NFSv3 can use UDP or TCP running over an IP. NFS V4 only works with TCP; UDP is no longer offered, so it is worth planning a transition to TCP now if you still use primarily NFS over UDP. With Data ONTAP® release 7.1 and higher, TCP connections are enabled automatically by default (Figure 1-9).

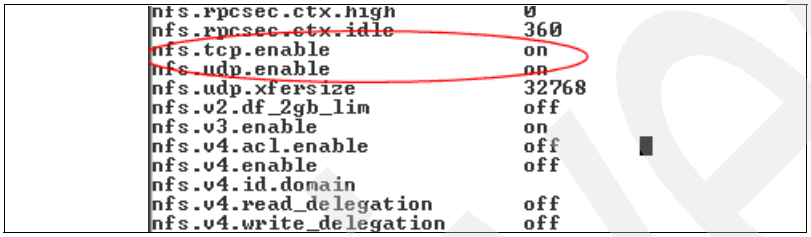


*Figure 1-9 Network transport protocol default setting on N series storage system*

NFS over TCP can handle multispeed networks (networks where the links connecting the server and the client use different speeds), higher levels of packet loss and congestion, fair bandwidth sharing, and widely varying network and server latency, but can cause long delays during server recovery. Although TCP has slightly greater network and CPU overhead on both the client and server, you will find that NFS performance on TCP remains stable across a variety of network conditions and workloads.

If you find UDP suits your needs better and you run kernels released previous to 2.4.22, be sure to enlarge your client's default socket buffer size by following the instructions listed in Appendix A.1, “Special network settings” on page 57 (also see Figure 1-10).

```
[itso@itso ~]$ cat /etc/sysctl.conf
# Kernel sysctl configuration file for Red Hat Linux
#
# For binary values, 0 is disabled, 1 is enabled.  See sysctl(8) and
# sysctl.conf(5) for more details.

# Controls IP packet forwarding
net.ipv4.ip_forward = 0

# Controls source route verification
net.ipv4.conf.default.rp_filter = 1

# Do not accept source routing
net.ipv4.conf.default.accept_source_route = 0

# Controls the System Request debugging functionality of the kernel
kernel.sysrq = 0

# Controls whether core dumps will append the PID to the core filename
# Useful for debugging multi-threaded applications.

kernel.semopm = 100
kernel.core_uses_pid = 1
kernel.shmmax = 2147483648
kernel.shmmni = 4096
kernel.shmall = 2097152
net.core.rmem_default=262144
net.core.wmem_default=262144
net.core.rmem_max=262144
net.core.wmem_max=262144
fs.file-max = 65536
net.ipv4.ip_local_port_range = 1024 65000
kernel.semopm = 100
#rmem_default=262144
#wmem_default=262144
#rmem_max=262144
#wmem_max=262144
kernel.sem=250 32000 100 128
[itso@itso ~]$ _
```

*Figure 1-10 Verify client's default send/receive socket buffer size*

Bugs in the IP fragmentation logic in these kernels can cause a client to flood the network with unusable packets, preventing other clients from accessing the N series products. The Linux NFS client is especially sensitive to IP fragmentation problems that can result from congested networks or undersized switch port buffers. If you think IP fragmentation is an issue for your clients using NFS over UDP, the **netstat -s** command on the Linux client (Figure 1-11) and on the N series storage system (Figure 1-12) will show continuous increases in the number of IP fragmentation errors. Be certain to apply this change to all Linux NFS clients on your network in order for the change to be completely effective. Note that RHEL 3.0's kernels, even though based on 2.4.21-pre1, already set transport socket buffer sizes correctly.

```
[itso@itso ~]$ netstat -s
Ip:
    58573 total packets received
    0 forwarded
    0 incoming packets discarded
    58573 incoming packets delivered
    4656 requests sent out
Icmp:
    174 ICMP messages received
    0 input ICMP message failed.
    ICMP input histogram:
        destination unreachable: 2
        echo requests: 120
        echo replies: 52
    131 ICMP messages sent
    0 ICMP messages failed
    ICMP output histogram:
        destination unreachable: 11
        echo replies: 120
Tcp:
    29 active connections openings
    10 passive connection openings
    0 failed connection attempts
    4 connection resets received
    1 connections established
    5440 segments received
    4419 segments send out
    4 segments retransmited
    0 bad segments received.
    2 resets sent
Udp:
    70 packets received
    11 packets to unknown port received.
    0 packet receive errors
    54 packets sent
```

*Figure 1-11 Monitoring IP fragmentation error on Linux client using netstat command*

```
itsotuc1>
itsotuc1> netstat -s -p ip
ip:
        91433 total packets received
        0 bad header checksums
        0 with size smaller than minimum
        0 with size larger than maximum
        0 with data size < data length
        0 with header length < data size
        0 with data length < header length
        0 with bad options
        0 with incorrect version number
        0 packets with spoofed source address
        0 packets arrived on wrong port
        0 fragments received
        the last 2 src addr that send fragments:
                0.0.0.0 0.0.0.0
        0 fragments dropped (dup or out of space)
        0 malformed fragments dropped
        0 overlapping fragments discarded
        0 fragments dropped after timeout
        the last 2 src addr that have fragement time out:
                0.0.0.0 0.0.0.0
        0 packets dropped, too many fragments
        the last 2 src addr that sent too many fragments:
                0.0.0.0 0.0.0.0
        0 packets dropped, reassembly queue overflow
        0 packets reassembled ok
        87299 packets for this host
        23 packets for unknown/unsupported protocol
        0 packets forwarded
        4111 packets not forwardable
        0 redirects sent
        6845 packets sent from this host
        0 packets sent with fabricated ip header
        0 output packets dropped due to no bufs, etc.
        0 output packets discarded due to no route
        0 output datagrams fragmented
        0 fragments created
        0 datagrams that can't be fragmented
itsotuc1>
```

*Figure 1-12 Monitoring IP fragmentation error*

**Note:** The kernel has a limit of how many fragments it can buffer before it starts throwing away packets. With kernels that support the /proc filesystem, you can monitor the files /proc/sys/net/ipv4/ipfrag_high_thresh and /proc/sys/net/ipv4/ipfrag_low_thresh. Once the number of fragmented packets reaches ipfrag_high_thresh (in bytes), the kernel will start dropping fragmented packets until the number of fragmented packets reaches the ipfrag_low_thresh (Figure 1-13).

Another counter to monitor is IP: ReasmFails in the file /proc/net/snmp. This is the number of fragment reassembly failures. If it goes up too quickly during heavy file activity, you may have a problem.

```
[itso@itso ~]$
[itso@itso ~]$ cat /proc/sys/net/ipv4/ipfrag_high_thresh
262144
[itso@itso ~]$ cat /proc/sys/net/ipv4/ipfrag_low_thresh
196608
[itso@itso ~]$ cat /proc/net/snmp
Ip: Forwarding DefaultTTL InReceives InHdrErrors InAddrErrors ForwDatagr
ams InUnknownProtos InDiscards InDelivers OutRequests OutDiscards OutNoR
outes ReasmTimeout ReasmReqds ReasmOKs ReasmFails FragOKs FragFails Frag
Creates
Ip: 2 64 59315 0 0 0 0 0 59315 4794 0 0 0 0 0 0 0 0 0
Icmp: InMsgs InErrors InDestUnreachs InTimeExcds InParmProbs InSrcQuench
s InRedirects InEchos InEchoReps InTimestamps InTimestampReps InAddrMask
s InAddrMaskReps OutMsgs OutErrors OutDestUnreachs OutTimeExcds OutParmP
robs OutSrcQuenchs OutRedirects OutEchos OutEchoReps OutTimestamps OutTi
mestampReps OutAddrMasks OutAddrMaskReps
Icmp: 174 0 2 0 0 0 0 120 52 0 0 0 0 131 0 11 0 0 0 0 0 120 0 0 0 0
Tcp: RtoAlgorithm RtoMin RtoMax MaxConn ActiveOpens PassiveOpens Attempt
Fails EstabResets CurrEstab InSegs OutSegs RetransSegs InErrs OutRsts
Tcp: 1 200 120000 -1 29 10 0 4 1 5663 4557 4 0 2
Udp: InDatagrams NoPorts InErrors OutDatagrams
Udp: 70 11 0 54
[itso@itso ~]$ _
```

*Figure 1-13 Monitoring IP fragmentation error on Linux client using /proc filesystem*

In Linux kernels prior to 2.4.21, a remote TCP disconnect (for example, during a cluster failover) occasionally may cause a deadlock on the client that makes a whole mount point unusable until the client is rebooted. There is no workaround other than upgrading to a version of the Linux kernel where this issue is addressed.

For databases, it is required to use NFS over TCP. This configuration is currently certified with Oracle9i and 10g RAC.

You can control RPC retransmission timeouts with the `timeo` option (Figure 1-7 on page 11). Retransmission is the mechanism by which clients ensure a server receives and processes an RPC request. If the client does not receive a reply for an RPC within a certain interval for any reason, it retransmits the request until it receives a reply from the server. After each retransmission, the client doubles the retransmit timeout up to 60 seconds to keep network load to a minimum.

By default, the client retransmits an unanswered UDP RPC request after 0.7 seconds. In general, it is not necessary to change the retransmission timeout for UDP, but in some cases, a shorter retransmission timeout for NFS over UDP may shorten latencies due to packet losses. As of kernel 2.4.20, an estimation algorithm that adjusts the timeout for optimal performance governs the UDP retransmission timeout for some types of RPC requests. Early versions of this estimator allowed extremely short retransmit timeouts, resulting in low performance and unnecessary error messages in the client's kernel log. The latest updates of RHEL 3 and SLES 8 contain fixes to address this problem.

The Linux NFS client quietly retransmits RPC requests several times before reporting in the kernel log that it has lost contact with an NFS server. You can control how many times the client retransmits the same request before reporting the loss using the `retrans` mount option.

Remember that whenever the hard mount option is in effect, an NFS client never gives up retransmitting an RPC until it gets a reply. Be careful not to use the similar-sounding `retry` mount option, which controls how long the mount command retries a backgrounded mount request before it gives up.

Retransmission for NFS over TCP works somewhat differently. The TCP network protocol contains its own timeout and retransmission mechanism that ensures packets arrive at the receiving end reliably and in order. The RPC client depends on this mechanism for recovering from the loss of RPC requests and thus uses a much longer timeout setting for NFS over TCP by default. Due to a bug in the **`mount`** command, the default retransmission timeout value on Linux for NFS over TCP is quite small, unlike other NFS client implementations. To obtain standard behavior, we strongly recommend using `timeo=600,retrans=2` (Figure 1-7 on page 11) explicitly when mounting via TCP. Unlike with NFS over UDP, using a short retransmission timeout with NFS over TCP does not have performance benefits and may increase the risk of data corruption.

In summary, we strongly recommend using TCP as the transport of choice for NFS on modern Linux distributions. To avoid IP fragmentation issues on both the client and filer, consider disabling NFS over UDP entirely on the N series product and explicitly specifying `tcp` on all your NFS mounts. In addition, we strongly recommend the explicit use of the `timeo=600` mount option on Linux to work around bugs in the **`mount`** command which shorten the retransmit timeout. If you must use NFS over UDP, ensure you adjust the size of transport socket buffers properly, and explicitly set a large number of retransmissions with the `retrans=` mount option.

## 1.5 Capping the size of read and write operations

NFS clients break application read and write requests into smaller chunks when communicating with NFS servers. The maximum size, in bytes, that a client uses for NFS read requests is called the *rsize*, and the maximum size a client uses for NFS write requests is called the *wsize*. Together, these two are often referred to as the *transfer size*, because there are few cases where the two need to have different values. On Linux, the maximum transfer size is 32KB, but the transfer size can be anywhere between 1KB and 32KB. The client ensures these values are rounded down to the nearest power of 2.

You can specify the transfer sizes explicitly when mounting an NFS server with the `rsize` and `wsize` mount options (Figure 1-14), or you can allow the client to choose a default value.

```
sunrpc /var/lib/nfs/rpc_pipefs rpc_pipefs rw 0 0
//192.168.3.249/temp /share smbfs rw,nodiratime,nosuid,nodev 0 0
192.168.3.254:/vol/oradata2 /mnt/oradata2 nfs rw,v3,rsize=32768,wsize=32
768,hard,tcp,lock,addr=192.168.3.254 0 0
[root@itso oradata1]#
[root@itso oradata1]# umount -t nfs 192.168.3.254:/vol/oradata2 /mnt/or
adata2
umount: /mnt/oradata2: not mounted
umount: /mnt/oradata2: not mounted
[root@itso oradata1]#
[root@itso oradata1]# mount -t nfs 192.168.3.254:/vol/oradata2 /mnt/ora
data2 -o rsize=8192,wsize=8192
[root@itso oradata1]#
[root@itso oradata1]#
[root@itso oradata1]# cat /proc/mounts
rootfs / rootfs rw 0 0
/proc /proc proc rw,nodiratime 0 0
none /dev tmpfs rw 0 0
/dev/root / ext3 rw 0 0
none /dev tmpfs rw 0 0
/proc /proc proc rw,nodiratime 0 0
/proc/bus/usb /proc/bus/usb usbfs rw 0 0
/sys /sys sysfs rw 0 0
none /dev/pts devpts rw 0 0
/dev/sdb1 /boot ext3 rw 0 0
none /dev/shm tmpfs rw 0 0
none /proc/sys/fs/binfmt_misc binfmt_misc rw 0 0
sunrpc /var/lib/nfs/rpc_pipefs rpc_pipefs rw 0 0
//192.168.3.249/temp /share smbfs rw,nodiratime,nosuid,nodev 0 0
192.168.3.254:/vol/oradata2 /mnt/oradata2 nfs rw,v3,rsize=8192,wsize=819
2,hard,tcp,lock,addr=192.168.3.254 0 0
[root@itso oradata1]#
```

*Figure 1-14 Changing transfer size of /vol/oradata2 using mount command and verifying the result*

The server and client take these values and negotiate the maximum allowed size based on what both client and server support. In fact, you can configure your N series storage system to limit the maximum transfer size using Data ONTAP's **`options nfs`** command to prevent clients from using a large transfer size (Figure 1-15 and Figure 1-16). Be careful not to change the maximum transfer size while there is outstanding read or write activity because some older Linux clients cannot recover gracefully from a change in the middle of a read or write operation.

```
itsotuc1>
itsotuc1> options nfs.tcp.xfersize
nfs.tcp.xfersize            32768
itsotuc1>
itsotuc1>
itsotuc1> options nfs.tcp.xfersize 65536
itsotuc1> options nfs.tcp.xfersize
nfs.tcp.xfersize            65536
itsotuc1>
```

*Figure 1-15 Adjusting the max TCP transfer size on N series storage system*

```
itsotuc1>
itsotuc1> options nfs.udp.xfersize
nfs.udp.xfersize            32768
itsotuc1> options nfs.udp.xfersize 57344
itsotuc1>
itsotuc1> options nfs.udp.xfersize
nfs.udp.xfersize            57344
itsotuc1>
```

*Figure 1-16 Adjusting the max UDP transfer size on N series storage system*

The network transport protocol (TCP or UDP) you choose interacts in complicated ways with the transfer size. When you encounter poor performance because of network problems, using NFS over TCP is a better way to achieve good performance than using small read or write sizes over UDP. The NFS client and server fragment large UDP datagrams, such as single read or write operations larger than the network's MTU, into individual IP packets. RPC over UDP retransmits a whole RPC request if any part of it is lost on the network, whereas RPC over TCP efficiently recovers a few lost packets and reassembles the complete request at the receiving end. Thus with NFS over TCP, 32KB read and write size usually provides good performance by allowing a single RPC to transmit or receive a large amount of data.

With NFS over UDP, 32KB read and write size may provide good performance, but using NFS over UDP often results in terrible performance if the network becomes congested. For Linux, a good compromise value when using NFS over UDP is to use a small multiple of the network's MTU. For example, typical Ethernet uses an MTU of 1524 bytes, so using a transfer size of 2048 or 4096 is probably going to give the best performance as network conditions vary over time. If you find even that does not work well, and you cannot improve network conditions, we recommend switching to NFS over TCP if possible.

By default, NFS client implementations choose the largest transfer size a server supports. However, if you do not explicitly set `rsize` or `wsize` when you mount an NFS filesystem on a 2.4-based Red Hat NFS client, the default value for both is a modest 4096 bytes. Red Hat chose this default because the default transport is UDP, and the small default transfer size setting allows the Linux NFS client to work without adjustment in most environments. Usually on clean high-performance networks, or with NFS over TCP, you can improve NFS performance by explicitly increasing these values.

In Linux, the `rsize` and `wsize` mount options have additional semantics compared with the same options as implemented in other operating systems. Normally, the Linux client caches application write requests, issuing NFS WRITE operations when it has at least *`wsize`* bytes to write. The NFS client often returns control to a writing application before it has issued any NFS WRITE operations. It also issues NFS READ operations in parallel before waiting for the server to reply to any of them. If rsize is set below the system's page size (4KB on x86 hardware), the NFS client in 2.4 kernels issues individual read operations one at a time and waits for each operation to complete before issuing the next read operation. If wsize is set below the system's page size, the NFS client issues synchronous writes without regard to the use of the *`sync`* or *`async`* mount options. As with reads, synchronous writes cause applications to wait until the NFS server completes each individual write operation before issuing the next operation or before letting an application continue with other processing. When performing synchronous writes, the client waits until the server has written its data to stable storage before allowing an application to continue.

Some hardware architectures allow a choice of different page sizes. Intel® Itanium® systems, for instance, support pages up to 64KB. On a system with 64KB pages, the rsize and wsize limitations described previously still apply; thus all NFS I/Os are synchronous on these systems, significantly slowing read and write throughput. This limitation has been removed in 2.6 kernels so that all read and write traffic is asynchronous whenever possible, independent of the transfer size settings. When running on hardware that supports different page sizes, choose a combination of page size and rsize/wsize that allows the NFS client to do asynchronous I/O if possible. Usually distributors choose a single large page size, such as 16KB, when they build kernels for hardware architectures that support multiple page sizes.

**Note:** The capabilities of the Linux NFS server are different from the capabilities of the Linux NFS client. As of the 2.4.21 kernel release, the Linux NFS server does not support NFS over TCP and does not support rsize and wsize larger than 8KB. The Linux NFS client, however, supports NFS over both UDP and TCP and rsize and wsize up to 32KB. Check with your Linux distributor to determine whether their kernels support serving files via NFS over TCP.

2

# Mount options

This chapter describes mount and unmount options that you may be need to implement depending on the situation or application.

## 2.1 Special mount options

Consider using the `bg` option (Figure 2-1)if your client system needs to be available even if it cannot mount some servers. This option causes mount requests to put themselves in the background automatically if a mount cannot complete immediately. When a client starts up and a server is not available, the client waits for the server to become available by default. The default behavior, which you can adjust with the *`retry`* mount option, results in waiting for almost a week before giving up.

```
[root@itso mnt]#
[root@itso mnt]# mount -t nfs 192.168.3.254:/vol/oradata1 /mnt/oradata1
 -o rw,bg,soft,nointr,tcp,nocto,noac,nosuid,retrans=2,timeo=600,rsize=6
5536,wsize=65536
[root@itso mnt]#
[root@itso mnt]# df -k
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/mapper/VolGroup00-LogVol00
                      32801652   6459116  24676284  21% /
/dev/sdb1               101086     12864     83003  14% /boot
none                    517328         0    517328   0% /dev/shm
//192.168.3.249/temp  29294080  13762560  15531520  47% /share
192.168.3.254:/vol/oradata1
                         32768     30880      1888  95% /mnt/oradata1
[root@itso mnt]#
[root@itso mnt]# cat /proc/mounts
rootfs / rootfs rw 0 0
/proc /proc proc rw,nodiratime 0 0
none /dev tmpfs rw 0 0
/dev/root / ext3 rw 0 0
none /dev tmpfs rw 0 0
/proc /proc proc rw,nodiratime 0 0
/proc/bus/usb /proc/bus/usb usbfs rw 0 0
/sys /sys sysfs rw 0 0
none /dev/pts devpts rw 0 0
/dev/sdb1 /boot ext3 rw 0 0
none /dev/shm tmpfs rw 0 0
none /proc/sys/fs/binfmt_misc binfmt_misc rw 0 0
sunrpc /var/lib/nfs/rpc_pipefs rpc_pipefs rw 0 0
//192.168.3.249/temp /share smbfs rw,nodiratime,nosuid,nodev 0 0
192.168.3.254:/vol/oradata1 /mnt/oradata1 nfs rw,sync,nosuid,v3,rsize=32
768,wsize=32768,acregmin=0,acregmax=0,acdirmin=0,acdirmax=0,soft,tcp,noc
to,noac,lock,addr=192.168.3.254 0 0
[root@itso mnt]#
```

*Figure 2-1 An example of mount using special mount options*

The `fg` option is useful when you need to serialize your mount requests during system initialization. For example, you probably want the system to wait for /usr to become available before proceeding with multiuser boot. If you mount /usr or other critical filesystems from an NFS server, you should consider using `fg` for these mounts. The `retry` mount option has no effect on foreground mounts. A foreground mount request will fail immediately without any retransmission if any problem occurs.

For security, you can also use the `nosuid` mount option(Figure 2-1). This causes the client to disable the special bits on files and directories. The Linux man page for the **`mount`** command recommends also disabling or removing the **`suidperl`** command when using this option. Note that the N series storage system also has a `nosuid` export option, which does roughly the same thing for all clients accessing an export. Interestingly, the storage system's `nosuid` export option also disables the creation of special devices. If you notice programs that use special sockets and devices (such as *screen*) behaving strangely, check for the `nosuid` export option on your N series storage system.

To enable Kerberos authentication on your NFS mounts, you can specify the *`sec=krb`* mount option. In addition to Kerberos authentication, you can also choose to enable authentication with request integrity checking (`sec=krb5i`), or authentication with privacy (`sec=krb5p`). Note that most Linux distributions do not yet support krb5p or other advanced security flavors such as SPKM3 or lipkey as of this writing. More on how to configure your NFS client to use NFS with Kerberos is described in 5.2, "Security" on page 50.

## 2.2 Tuning NFS client cache behavior

Other mount options allow you to tailor the client's attribute caching and retry behavior. It is not necessary to adjust these behaviors under most circumstances. However, sometimes you must adjust NFS client behavior to make NFS appear to your applications more like a local filesystem, or to improve performance for metadata-intensive workloads.

There are a few indirect ways to tune client-side caching. First, the most effective way to improve client-side caching is to add more RAM to your clients. Linux will make appropriate use of the new memory automatically. To determine how much RAM you need to add, determine how large your active file set is and increase RAM accordingly. This greatly reduces cache turnover rate. You should see fewer read requests and faster client response time as a result.

Some working sets will never fit in a client's RAM cache. Your clients may have 128MB or 4GB of RAM, for example, but you may still see significant client cache turnover. In this case, reducing *cache miss* latency is the best approach. You can do this by improving your network infrastructure and tuning your server to improve its performance. Because a client-side cache is not effective in these cases, you may find that keeping the client's cache small is beneficial.

Normally, for each file in a filesystem that has been accessed recently, a client caches file attribute information, such as a file's last modification time (mtime) and size. To detect file changes quickly yet efficiently, the NFS protocol uses *close-to-open* cache semantics. When a client opens a file, it uses a GETATTR operation to check that the file still exists and any cached data it has is still up-to-date. A client checks back with the server only after a timeout indicates that the file's attributes may be stale. During such a check, if the server's version of the attributes has changed, the client purges its cache. A client can delay writes to a file indefinitely. When a client closes a file, however, it flushes all pending modifications to the file to the server. This allows a client to provide good performance in most cases, but means it might take some time before an application running on one client sees changes made by applications on other clients.

Clients should check back with the server every so often to be sure cached attribute information is still valid. However, adding RAM on the client will not affect the rate at which the client tries to revalidate parts of the directory structure it has already cached. No matter how much of the directory structure is cached on the client, it must still validate what it knows when files are opened or when attribute cache information expires. You can lengthen the attribute cache timeout with the `actimeo` mount option to reduce the rate at which the client tries to revalidate its attribute cache. With the 2.4.19 kernel release, you can also use the `nocto` mount option to reduce the revalidation rate even further, at the expense of cache coherency among multiple clients. The `nocto` mount option is appropriate for read-only mount points where files change infrequently, such as a lib, include, or bin directory, static HTML files, or image libraries. In combination with judicious settings of `actimeo` you can significantly reduce the number of on-the-wire operations generated by your NFS clients. Be careful to test this setting with your application to be sure that it will tolerate the delay before the NFS client notices file changes made by other clients and fetches the new versions from the server.

Certain distributions also support an option that disables Access Control List support on NFSv3 mounts. After update 3, RHEL 3 allows you to disable ACL support, eliminating all ACCESS operations on the wire for a mount point. ACCESS operations are synchronous, slowing applications unnecessarily if they do not use or support ACLs. Using the `noacl` mount option is safe when your files are stored on N series storage systems that are accessed only by NFS version 2 or 3.

**Note:** Do not use the `noacl` mount option if your files reside on a volume that is being shared by other types of clients that have native ACL support, such as CIFS or NFSv4 client.

The Linux NFS client delays application writes to combine them into larger, more efficiently processed requests. You can guarantee that a client immediately pushes every write system call that an application makes to servers by using the `sync` mount option. This is useful when an application needs the guarantee that data is safe on disk before it continues. Frequently such applications already use the `O_SYNC` open flag or invoke the flush system call when needed. Thus, the `sync` mount option is often not necessary.

Delayed writes and the client's attribute cache timeout can delay detection of changes on the server by many seconds while a file is open. The `noac` mount option prevents the client from caching file attributes. This means that every file operation on the client that requires file attribute information results in a GETATTR operation to retrieve a file's attribute information from the server. Note that `noac` also causes a client to process all writes to that filesystem synchronously, just as the `sync` mount option does. Disabling attribute caching is only one part of `noac`; it also guarantees that data modifications are visible on the server so that other clients using `noac` can detect them immediately. Thus, `noac` is shorthand for `actimeo=0,sync`.

When the `noac` option is in effect, clients still cache file data as long as they detect that a file has not changed on the server. The `noac` mount option allows a client to keep very close track of files on a server so it can discover changes made by other clients quickly. Normally you will not use this option, but it is important when an application that depends on single system behavior is deployed across several clients.

Using the `noac` mount option causes a 40% performance degradation on typical workloads, but some common workloads, such as sequential write workloads, can be impacted by up to 70%. Database workloads that consist of random reads and writes are generally less affected by using `noac`. `Noac` mount option generates a very large number of GETATTR operations and sends write operations synchronously. Both of these add significant protocol overhead. The `noac` mount option trades off single-client performance for client cache coherency. Only applications that need tight cache coherency among multiple clients require that filesystems be mounted with the `noac` mount option.

Some applications require direct, uncached access to data on a server. Using the `noac` mount option is sometimes not good enough, because even with this option, the Linux NFS client still caches reads. To ensure your application uses the server's version of a file's data and not potentially stale data cached by the client, your application can lock and unlock the file. This pushes all pending write operations back to the server and purges any remaining cached data, so the next read operation will go back to the server rather than reading from a local cache.

Alternatively, the Linux NFS client in the RHEL and SLES kernels supports direct I/O to NFS files when an application opens a file with the `O_DIRECT` flag. Direct I/O is a feature designed to benefit database applications that manage their own data cache. When this feature is enabled, an application's read and write system calls are translated directly into NFS read and write operations. The Linux kernel never caches the results of any read or write when a file is opened with this flag, so applications always get exactly what is on the server. Because of I/O alignment restrictions in some versions of the Linux `O_DIRECT` implementation, applications must be modified to support direct I/O properly. See Appendix A.4, "Enabling uncached I/O on RHEL AS 2.1" on page 61 for more information on this feature, and its equivalent in RHEL AS 2.1, uncached I/O.

For some servers or applications, it is necessary to prevent the Linux NFS client from sending Network Lock Manager requests. You can use the `nolock` mount option to prevent the Linux NFS client from notifying the server's lock manager when an application locks a file. Note, however, that the client still flushes its data cache and uses more restrictive write back semantics when a file lock is in effect. The client always flushes all pending writes whenever an application locks or unlocks a file.

## 2.3 Mounting with NFS version 4

RHEL 4.0 and 2.6-based Fedora Core distributions introduce support for the latest version of NFS, version 4. In this section we discuss how to use NFSv4 on your Linux NFS client, and what mount options your client supports. You should ensure your filer is running Data ONTAP 7.1 or newer before trying NFSv4.

The Linux NFS client now recognizes two different filesystem types:

- The *nfs* filesystem type uses the `vers=` mount option to determine whether to use NFSv2 or NFSv3 when communicating with a server.
- The *nfs4* filesystem type supports only NFSv4 and does not recognize the `vers=` mount option.

If you have scripts that specify particular filesystem types to act on NFS filesystems, you will need to modify them to work with both nfs and nfs4 filesystems.

The NFSv4 wire protocol represents users and groups as strings instead of UIDs and GIDs. Before attempting to mount with NFSv4, you must ensure that the client's UID mapper is configured and running. Otherwise, the client will map all UIDs and GIDs to the user *nobody*. The mapper's configuration file is /etc/idmap.conf (Figure 2-2).

```
none /proc/sys/fs/binfmt_misc binfmt_misc rw 0 0
sunrpc /var/lib/nfs/rpc_pipefs rpc_pipefs rw 0 0
//192.168.3.249/temp /share smbfs rw,nodiratime,nosuid,nodev 0 0
192.168.3.254:/vol/oradata2 /mnt/oradata2 nfs rw,v3,rsize=8192,wsize=819
2,hard,tcp,lock,addr=192.168.3.254 0 0
[root@itso oradata1]#
[root@itso oradata1]#
[root@itso oradata1]#
[root@itso oradata1]# cat /etc/idmap.conf
[General]

Verbosity = 0
Pipefs-Directory = /var/lib/nfs/rpc_pipefs
Domain = localdomain

[Mapping]

Nobody-User = nobody
Nobody-Group = nobody

[Translation]
Method = nsswitch
[root@itso oradata1]#
[root@itso oradata1]# ls -lt
total 0
-rw-r--r--  1 root root 0 Dec 14 10:34 test
[root@itso oradata1]#
[root@itso oradata1]# /etc/init.d/rpcidmapd start
Starting RPC idmapd:                       OK  ]
```

*Figure 2-2 An example of idmap.conf*

Typically the only change needed is to specify the real domain name of the client so that it can map local UIDs and GIDs on the client to network names. When this is done, start the UID mapper daemon with this command:

```
/etc/init.d/rpcidmapd start
```

To mount a server that supports NFS version 4, you can use the following command:

```
mount –t nfs4 –o rw,rsize=32768,wsize=32768 filer:/vol/vol0 /mnt/nfs
```

## Mount option changes in NFSv4

Some mount options you may be accustomed to using with NFS versions 2 and 3 are no longer supported with the nfs4 filesystem type.

- *vers=* is not supported. The `udp` and `tcp` mount options are no longer supported; instead, use `proto=` if you would like to choose a transport that is not TCP. N series storage systems follow the NFSv4 specification and do not support NFSv4 on UDP, so `proto=tcp` is the only protocol you can use with NFSv4 when your clients communicate with N series storage systems.
- Mount options `noacl`, `nocto`, and `nolock` are no longer available with NFSv4.
- To avoid possible delegation issue, make sure nfs.v4.read_delegation and nfs.v4.write_delegation are disabled on your N series storage system using the Data ONTAP **`options`** command (Example 2-1) if you plan to access the N series storage system via NFSv4 from a Linux NFS client (Figure 2-3).

*Example 2-1 Disabling read and write delegation NFSv4 options*

```
options nfs.v4.read_delegation off
options nfs.v4.write_delegation off
```

```
nfs.ifc.xmt.high              16
nfs.ifc.xmt.low               8
nfs.kerberos.enable           off
nfs.kerberos.file_keytab.enable off
nfs.locking.check_domain      on
nfs.mount_rootonly            off
nfs.mountd.trace              off
nfs.netgroup.strict           off
nfs.notify.carryover          on
nfs.per_client_stats.enable   off
nfs.require_valid_mapped_uid  off
nfs.response.trace            off
nfs.response.trigger          60
nfs.rpcsec.ctx.high           0
nfs.rpcsec.ctx.idle           360
nfs.tcp.enable                on
nfs.tcp.xfersize              65536
nfs.udp.enable                on
nfs.udp.xfersize              57344
nfs.v2.df_2gb_lim             off
nfs.v3.enable                 on
nfs.v4.acl.enable             off
nfs.v4.enable                 off
nfs.v4.id.domain              localhost.localdomain
nfs.v4.read_delegation        off
nfs.v4.write_delegation       off
nfs.webnfs.enable             off
nfs.webnfs.rootdir            XXX
nfs.webnfs.rootdir.set        off
itsotuc1>
itsotuc1> options nfs.v4
nfs.v4.acl.enable             off
nfs.v4.enable                 off
nfs.v4.id.domain              localhost.localdomain
nfs.v4.read_delegation        off
nfs.v4.write_delegation       off
itsotuc1> _
```

*Figure 2-3 Disabling NFSv4 read_ and write_delegation options using options command*

- The Linux NFS version 4 client also does not fully support file migration and replication via FS_LOCATIONS, nor does it fully support crossing security boundaries (going from, say, a volume exported with `sec=sys` to a volume exported with `sec=krb5`) automatically.

## 2.4 Unmounting NFS filesystems

This section discusses the unique subtleties of unmounting NFS filesystems on Linux.

Like other *NIX operating systems, the **umount** command detaches one or more filesystems from a client's filesystem hierarchy (like several other common features of *NIX, the name of this command is missing a letter). Normally, you use **umount** with no options, specifying only the path to the root of the filesystem you want to unmount.

Sometimes you might want more than a standard unmount operation, for example, when programs appear to be waiting indefinitely for access to files in a filesystem or if you want to clear the client's data cache.

- To unmount all currently mounted NFS filesystems described in /etc/mtab, issue the following command (Figure 2-4):

  ```
  umount –a –t nfs
  ```

  **Note:** A separate command is necessary to unmount NFSv4 filesystems:

  ```
  umount –a –t nfs4
  ```

- To unmount an NFS filesystem that becomes unresponsive, there are two **umount** options:
  - The following option forces an unmount operation to occur if the mounted NFS server is not reachable, as long as there are no RPC requests on the client waiting for a reply (Figure 2-5):

    ```
    umount –f path/to/filesystem/root
    ```

  - If using Linux kernel later than 2.4.11, the following option usually causes the Linux kernel to detach a filesystem from the client's filesystem hierarchy immediately, but allows it to clean up RPC requests and open files in the background (Figure 2-6):

    ```
    umount –l path/to/filesystem/root
    ```

```
ot@itso ~]# cat /proc/mounts
tfs / rootfs rw 0 0
oc /proc proc rw,nodiratime 0 0
e /dev tmpfs rw 0 0
v/root / ext3 rw 0 0
e /dev tmpfs rw 0 0
oc /proc proc rw,nodiratime 0 0
oc/bus/usb /proc/bus/usb usbfs rw 0 0
s /sys sysfs rw 0 0
e /dev/pts devpts rw 0 0
v/sdb1 /boot ext3 rw 0 0
e /dev/shm tmpfs rw 0 0
e /proc/sys/fs/binfmt_misc binfmt_misc rw 0 0
rpc /var/lib/nfs/rpc_pipefs rpc_pipefs rw 0 0
92.168.3.249/temp /share smbfs rw,nodiratime,nosuid,nodev 0 0
.168.3.254:/vol/oradata1 /mnt/oradata1 nfs rw,nosuid,v3,rsize=32768,wsize=
hard,tcp,lock,addr=192.168.3.254 0 0
.168.3.254:/vol/oradata2 /mnt/oradata2 nfs rw,nosuid,v3,rsize=32768,wsize=
hard,tcp,lock,addr=192.168.3.254 0 0
ot@itso ~]#
ot@itso ~]# umount -a -t nfs
ot@itso ~]#
ot@itso ~]# cat /proc/mounts
tfs / rootfs rw 0 0
oc /proc proc rw,nodiratime 0 0
e /dev tmpfs rw 0 0
v/root / ext3 rw 0 0
e /dev tmpfs rw 0 0
oc /proc proc rw,nodiratime 0 0
oc/bus/usb /proc/bus/usb usbfs rw 0 0
s /sys sysfs rw 0 0
e /dev/pts devpts rw 0 0
v/sdb1 /boot ext3 rw 0 0
e /dev/shm tmpfs rw 0 0
e /proc/sys/fs/binfmt_misc binfmt_misc rw 0 0
rpc /var/lib/nfs/rpc_pipefs rpc_pipefs rw 0 0
92.168.3.249/temp /share smbfs rw,nodiratime,nosuid,nodev 0 0
ot@itso ~]# _
```

*Figure 2-4 An example of Linux utility umount -a command*

```
[root@itso ~]# cat /proc/mounts
rootfs / rootfs rw 0 0
/proc /proc proc rw,nodiratime 0 0
none /dev tmpfs rw 0 0
/dev/root / ext3 rw 0 0
none /dev tmpfs rw 0 0
/proc /proc proc rw,nodiratime 0 0
/proc/bus/usb /proc/bus/usb usbfs rw 0 0
/sys /sys sysfs rw 0 0
none /dev/pts devpts rw 0 0
/dev/sdb1 /boot ext3 rw 0 0
none /dev/shm tmpfs rw 0 0
none /proc/sys/fs/binfmt_misc binfmt_misc rw 0 0
sunrpc /var/lib/nfs/rpc_pipefs rpc_pipefs rw 0 0
//192.168.3.249/temp /share smbfs rw,nodiratime,nosuid,nodev 0 0
192.168.3.254:/vol/oradata2 /mnt/oradata2 nfs rw,v3,rsize=32768,wsize=3276
,tcp,lock,addr=192.168.3.254 0 0
[root@itso ~]#
[root@itso ~]# umount -f /mnt/oradata2
[root@itso ~]#
[root@itso ~]# cat /proc/mounts
rootfs / rootfs rw 0 0
/proc /proc proc rw,nodiratime 0 0
none /dev tmpfs rw 0 0
/dev/root / ext3 rw 0 0
none /dev tmpfs rw 0 0
/proc /proc proc rw,nodiratime 0 0
/proc/bus/usb /proc/bus/usb usbfs rw 0 0
/sys /sys sysfs rw 0 0
none /dev/pts devpts rw 0 0
/dev/sdb1 /boot ext3 rw 0 0
none /dev/shm tmpfs rw 0 0
none /proc/sys/fs/binfmt_misc binfmt_misc rw 0 0
sunrpc /var/lib/nfs/rpc_pipefs rpc_pipefs rw 0 0
//192.168.3.249/temp /share smbfs rw,nodiratime,nosuid,nodev 0 0
[root@itso ~]# _
```

*Figure 2-5 An example of umount -f command*

```
[root@itso ~]# cat /proc/mounts
rootfs / rootfs rw 0 0
/proc /proc proc rw,nodiratime 0 0
none /dev tmpfs rw 0 0
/dev/root / ext3 rw 0 0
none /dev tmpfs rw 0 0
/proc /proc proc rw,nodiratime 0 0
/proc/bus/usb /proc/bus/usb usbfs rw 0 0
/sys /sys sysfs rw 0 0
none /dev/pts devpts rw 0 0
/dev/sdb1 /boot ext3 rw 0 0
none /dev/shm tmpfs rw 0 0
none /proc/sys/fs/binfmt_misc binfmt_misc rw 0 0
sunrpc /var/lib/nfs/rpc_pipefs rpc_pipefs rw 0 0
//192.168.3.249/temp /share smbfs rw,nodiratime,nosuid,nodev 0 0
192.168.3.254:/vol/oradata2 /mnt/oradata2 nfs rw,nosuid,v3,rsize=32768,wsi
68,hard,tcp,lock,addr=192.168.3.254 0 0
192.168.3.254:/vol/oradata1 /mnt/oradata1 nfs rw,nosuid,v3,rsize=32768,wsi
68,hard,tcp,lock,addr=192.168.3.254 0 0
[root@itso ~]#
[root@itso ~]# umount -l /mnt/oradata2
[root@itso ~]#
[root@itso ~]# cat /proc/mounts
rootfs / rootfs rw 0 0
/proc /proc proc rw,nodiratime 0 0
none /dev tmpfs rw 0 0
/dev/root / ext3 rw 0 0
none /dev tmpfs rw 0 0
/proc /proc proc rw,nodiratime 0 0
/proc/bus/usb /proc/bus/usb usbfs rw 0 0
/sys /sys sysfs rw 0 0
none /dev/pts devpts rw 0 0
/dev/sdb1 /boot ext3 rw 0 0
none /dev/shm tmpfs rw 0 0
none /proc/sys/fs/binfmt_misc binfmt_misc rw 0 0
sunrpc /var/lib/nfs/rpc_pipefs rpc_pipefs rw 0 0
//192.168.3.249/temp /share smbfs rw,nodiratime,nosuid,nodev 0 0
192.168.3.254:/vol/oradata1 /mnt/oradata1 nfs rw,nosuid,v3,rsize=32768,wsi
68,hard,tcp,lock,addr=192.168.3.254 0 0
```

*Figure 2-6 An example of umount -l command*

As mentioned previously, unmounting an NFS filesystem does not interrupt RPC requests that are awaiting a server reply. If an **umount** command fails because processes are waiting for network operations to finish, you must interrupt each waiting process using ^C or an appropriate **kill** command. Stopping stuck processes usually happens automatically during system shutdown so that NFS filesystems can be safely unmounted.

To identify processes waiting for NFS operations to complete, use the **lsof** (LiSt Open File) command, This is a Linux utility that allows you to view current network connections and the files associated with them (Figure 2-7). It performs similar function as other utilities like netstat and fuser, and allows you to find specific information on ports, users, processes, and files.

To list all open files corresponding to all active connections:

```
lsof
```

```
[root@itso mnt]# lsof |more
COMMAND    PID     USER   FD      TYPE     DEVICE     SIZE       NODE NAM
E
init         1     root  cwd       DIR      253,0     4096          2 /
init         1     root  rtd       DIR      253,0     4096          2 /
init         1     root  txt       REG      253,0    32684    2142172 /sb
in/init
init         1     root  mem       REG      253,0    56320     197749 /li
b/libselinux.so.1
init         1     root  mem       REG      253,0   106397     197732 /li
b/ld-2.3.4.so
init         1     root  mem       REG      253,0  1454546     197733 /li
b/tls/libc-2.3.4.so
init         1     root  mem       REG      253,0    53736     196313 /li
b/libsepol.so.1
init         1     root   10u      FIFO       0,13              1021 /de
v/initctl
migration    2     root  cwd       DIR      253,0     4096          2 /
migration    2     root  rtd       DIR      253,0     4096          2 /
migration    2     root  txt   unknown                               /pr
oc/2/exe
ksoftirqd    3     root  cwd       DIR      253,0     4096          2 /
ksoftirqd    3     root  rtd       DIR      253,0     4096          2 /
ksoftirqd    3     root  txt   unknown                               /pr
oc/3/exe
migration    4     root  cwd       DIR      253,0     4096          2 /
migration    4     root  rtd       DIR      253,0     4096          2 /
migration    4     root  txt   unknown                               /pr
oc/4/exe
ksoftirqd    5     root  cwd       DIR      253,0     4096          2 /
ksoftirqd    5     root  rtd       DIR      253,0     4096          2 /
ksoftirqd    5     root  txt   unknown                               /pr
oc/5/exe
events/0     6     root  cwd       DIR      253,0     4096          2 /
events/0     6     root  rtd       DIR      253,0     4096          2 /
events/0     6     root  txt   unknown                               /pr
oc/6/exe
events/1     7     root  cwd       DIR      253,0     4096          2 /
```

*Figure 2-7 An example of Linux utility lsof*

To obtain a list of all available options (Figure 8):

```
lsof -h
```

```
[root@itso ~]# lsof -h
lsof 4.72
 latest revision: ftp://lsof.itap.purdue.edu/pub/tools/unix/lsof/
 latest FAQ: ftp://lsof.itap.purdue.edu/pub/tools/unix/lsof/FAQ
 latest man page: ftp://lsof.itap.purdue.edu/pub/tools/unix/lsof/lsof_man
 usage: [-?abhlnNoOPRstUvV] [+|-c c] [+|-d s] [+D D] [+|-f]
  [-F [f]] [-g [s]] [-i [i]] [+|-L [l]] [+m [m]] [+|-M] [-o [o]]
[-p s] [+|-r [t]] [-S [t]] [-T [t]] [-u s] [+|-w] [-x [fl]] [--] [names]
Defaults in parentheses; comma-separate set (s) items; dash-separate ranges.
  -?|-h list help          -a AND selections (OR)      -b avoid kernel blocks
  -c c  cmd c, /c/[bix]    +c w  COMMAND width (9)
  +d s  dir s files        -d s  select by FD set      +D D  dir D tree *SLOW?*
                           -i select IPv[46] files     -l list UID numbers
  -n no host names         -N select NFS files         -o list file offset
  -O avoid overhead *RISK  -P no port names            -R list paRent PID
  -s list file size        -t terse listing            -T disable TCP/TPI info
  -U select Unix socket    -v list version info        -V verbose search
  +|-w  Warnings (+)       -- end option scan
  +f|-f  +filesystem or -file names
  -F [f] select fields; -F? for help
  +|-L [l] list (+) suppress (-) link counts < l (0 = all; default = 0)
                                        +m [m] use|create mount supplement
  +|-M   portMap registration (-)       -o o   o 0t offset digits (8)
  -p s   select by PID set              -S [t] t second stat timeout (15)
  -T qs TCP/TPI Q,St (s) info
  -g [s] select by process group ID set and print process group IDs
  -i i   select by IPv[46] address: [46][proto][@host|addr][:svc_list|port_list
  +|-r [t] repeat every t seconds (15); + until no files, - forever
  -u s   exclude(^)|select login|UID set s
  -x [fl] cross over +d|+D File systems or symbolic Links
  names  select named files or files on named file systems
Anyone can list all files; /dev warnings disabled; kernel ID check disabled.
[root@itso ~]# _
```

*Figure 8 An example of Linux utility lsof -h*

To list all open files associated with Internet connections (Figure 2-9):

```
lsof -i
```

```
ot@itso ~]#
ot@itso ~]# lsof -i
MAND     PID     USER   FD   TYPE DEVICE SIZE NODE NAME
tmap    2019      rpc    3u  IPv4   4782       UDP *:sunrpc
tmap    2019      rpc    4u  IPv4   4785       TCP *:sunrpc (LISTEN)
.statd  2039 rpcuser    4u  IPv4   4822       UDP *:1024
.statd  2039 rpcuser    5u  IPv4   4811       UDP *:943
.statd  2039 rpcuser    6u  IPv4   4825       TCP *:32770 (LISTEN)
d       2222     root   3u  IPv6   5211       TCP *:ssh (LISTEN)
etd     2237     root   5u  IPv4   5363       TCP *:ftp (LISTEN)
etd     2237     root   6u  IPv4   5364       TCP *:telnet (LISTEN)
dmail   2257     root   3u  IPv4   5314       TCP itso:smtp (LISTEN)
remote  2379     root   3u  IPv4   5828       UDP *:860
remote  2379     root   4u  IPv4   5831       TCP *:862 (LISTEN)
iserv.  2515     root   3u  IPv4   6011       TCP *:10000 (LISTEN)
iserv.  2515     root   5u  IPv4   6012       UDP *:10000
sd      3786     root   0u  IPv4   9425       TCP itso:ipp (LISTEN)
sd      3786     root   2u  IPv4   9426       UDP *:ipp
netd    5812     root   0u  IPv4  16091       TCP itso:telnet->192.168.3.242:3
 (ESTABLISHED)
netd    5812     root   1u  IPv4  16091       TCP itso:telnet->192.168.3.242:3
 (ESTABLISHED)
netd    5812     root   2u  IPv4  16091       TCP itso:telnet->192.168.3.242:3
 (ESTABLISHED)
```

*Figure 2-9 An example of Linux utility lsof -i*

**Note:** The NFS client allows these methods to interrupt (kill) pending RPC requests only if the mount option `intr` is set for that filesystem.

The **`umount`** command accepts filesystem specific options via the `-o` flag. The NFS client does not have any special options.

## 2.5 Mount option examples

We provide the following examples as a basis for beginning your experimentation. Start with an example that closely matches your scenario, then thoroughly test the performance and reliability of your application while refining the mount options you have selected.

- On older Linux systems, if you do not specify any mount options, the Linux mount command (or the automounter) automatically chooses these defaults:

  ```
  mount –o rw,fg,vers=2,udp,rsize=4096,wsize=4096,hard,intr,
  timeo=7,retrans=5
  ```

  - These default settings are designed to make NFS work right out of the box in most environments.
  - Almost every NFS server supports NFS version 2 over UDP.
  - `Rsize` and `wsize` are relatively small because some network environments fragment large UDP packets, which can hurt performance if there is a chance that fragments can be lost.
  - The RPC retransmit timeout is set to 0.7 seconds by default to accommodate slow servers and networks.

On clean single-speed networks, these settings are unnecessarily conservative. Over some firewalls, UDP packets larger than 1536 are not supported, so these settings do not work. On congested networks, UDP may have difficulty recovering from large numbers of dropped packets. NFS version 2 write performance is usually slower than NFS version 3. As you can see, there are many opportunities to do better than the default mount options.

- On those newer Linux distributions, the default mount options are set more reasonably, as follows:

  ```
  mount –o rw,bg,vers=3,tcp,timeo=600,rsize=32768,wsize=32768,hard,intr
  ```

  - The `bg` option means our client will be able to finish booting without waiting for N series storage systems that may be unavailable because of network outages.
  - The `hard` option minimizes the likelihood of data loss during network and server instability, while `intr` allows users to interrupt applications that may be waiting for a response from an unavailable server.
  - The `tcp` option works well on many typical LANs with 32KB read and write size.
  - The `timeo=600` is a good default for TCP mounts, but for UDP mounts, `timeo=4` might be more appropriate.
- Here is an example of a poor combination of mount options:

  ```
  mount –o rw,soft,udp,rsize=1024,wsize=1024
  ```

  - Using `soft` mounts with UDP is a recipe for silent data corruption. On Linux 2.4 with these mount options it is made worse.
  - The `wsize=1024` mount option on Linux 2.4 mandates synchronous writes; writes go to the server one at a time rather than in groups, and the client requires the server to push data onto disk before responding to the client's write operation request. If a server gathers writes to improve disk bandwidth, it delays its response to each write request waiting for more write requests, which can trigger the client to retry write requests. A server that delays responding to writes as long as several hundred milliseconds will probably cause the client to drop requests unnecessarily after several retries (note that N series storage systems usually do not delay writes because they can cache these requests in nonvolatile RAM to fulfill NFSv2's requirement for stable writes).
  - To address these issues, always use the `hard` option and use read and write sizes larger than your client's page size.
- When mounting a group of home directories over a WAN, you could try these mount options:

  ```
  mount –o rw,bg,vers=3,nosuid,tcp,timeo=600,retrans=2,rsize=2048,
  wsize=2048,soft,intr
  ```

  - This example uses NFS over TCP because NFS clients often reside on slower, less capable networks than servers. In this case, the TCP protocol can provide fast recovery from packet losses caused by network speed transitions and noisy phone lines. Using the `nosuid` mount option means users cannot create or use suid programs that reside in their home directories, providing a certain degree of safety from Trojan horses.
  - Limiting the maximum size of read and write operations gives interactive sessions on slow network links an advantage by keeping very large packets off the wire. On fast networks, large `rsize` and `wsize` values, such as 32768, are more appropriate.
  - The `soft` option helps to recover quickly from server or network outages with a minimal risk of possible data loss.
  - The `timeo=600` option allows the TCP protocol a long time to attempt recovery before the RPC client interferes.
- When mounting an N series storage system from an anonymous FTP or HTTP server, you could use the following mount options:

  ```
  mount –o ro,fg,vers=3,tcp,timeo=600,retrans=2,rsize=32768,wsize=32768,
  hard,nointr,nocto,actimeo=600
  ```

- The `fg` option ensures that NFS files are available before the FTP or HTTP server is started.
- The `ro` option anticipates that the FTP or HTTP server will never write data into files.
- The `nocto` option helps reduce the number of GETATTR and LOOKUP operations at the expense of tight cache coherency with other clients. The FTP server will see changes to files on the server after its attribute cache times out (usually after about one minute). Lengthening the attribute cache timeout also reduces the attribute cache revalidation rate.

- When mounting an N series storage system for use with a single-instance Oracle database system over a clean Gigabit Ethernet, you might try the following mount options (also shown in Figure 2-10):

  `mount –o rw,fg,vers=3,tcp,timeo=600,retrans=2,hard,nointr`

  - The `fg` option ensures that NFS filesystems are available before the database instance starts up.
  - We use `tcp` here because even though the physical network is fast and clean, TCP adds extra data integrity guarantees.
  - The `hard` option ensures data integrity in the event of network problems or a cluster failover event.
  - The `nointr` option prevents signals from interrupting NFS client operations. Such interruptions may occur during a shutdown abort, for instance, and are known to cause database corruption. File locking should be enabled when running databases in production as a degree of protection against corruption caused by improper backup procedures (for example, another instance of the same database running at a disaster recovery site against the same files as your normal production instance).

```
[root@itso ~]# mount -t nfs -o rw,fg,vers=3,tcp,timeo=600,retrans=2,hard,nointr
 192.168.3.254:/vol/oradata2 /mnt/oradata2
[root@itso ~]#
[root@itso ~]# cat /etc/mtab
/dev/mapper/VolGroup00-LogVol00 / ext3 rw 0 0
none /proc proc rw 0 0
none /sys sysfs rw 0 0
none /dev/pts devpts rw,gid=5,mode=620 0 0
usbfs /proc/bus/usb usbfs rw 0 0
/dev/sdb1 /boot ext3 rw 0 0
none /dev/shm tmpfs rw 0 0
none /proc/sys/fs/binfmt_misc binfmt_misc rw 0 0
sunrpc /var/lib/nfs/rpc_pipefs rpc_pipefs rw 0 0
//192.168.3.249/temp /share smbfs 0 0 0
192.168.3.254:/vol/oradata1 /mnt/oradata1 nfs rw,addr=192.168.3.254 0 0
192.168.3.254:/vol/oradata2 /mnt/oradata2 nfs rw,fg,nfsvers=3,tcp,timeo=600,retr
ans=2,hard,nointr,addr=192.168.3.254 0 0
```

*Figure 2-10 Mounting option for use with a single-instance Oracle database system over a clean Gigabit Ethernet*

See Appendix A.2, “Controlling file read-ahead in Linux” on page 60 for more information on how to set up databases on Linux client, including details on how to adjust the Linux read-ahead algorithm.

You are strongly advised to use NFSv3 and NFS over TCP wherever possible. If NFS over UDP is slow or hangs, this is a sign of network problems. Try to use TCP instead. Avoid using the `soft` mount option. Try the special mount options if you need an extra boost in performance.

3

# Performance

This chapter covers aspects of Linux client performance, with a special focus on networking.

## 3.1 Linux NFS client performance

The Linux NFS client runs in many different environments, from light desktop usage to databases with a dedicated private SAN. In general, the Linux NFS client can perform as well as most other type of NFS clients, and better than some, in these environments. However, you *must* plan your mount options and observe network behavior carefully to ensure the Linux NFS client performs at its best.

If your clients use high-performance networking (gigabit or faster), keep the following considerations in mind:

- You should plan to provide enough CPU and memory bandwidth on your clients to handle the interrupt and data rate.
- The NFS client software and the gigabit driver cut into resources available to user-level applications, so make sure there are enough resources to go around.
- Most gigabit cards that support 64-bit PCI or better should provide good performance. For most purposes, Gigabit Ethernet over copper works about as well as Gigabit Ethernet over fiber for short distance links.
- Category 5E or Category 6 cables are necessary for reliable performance on copper links. Fiber adds long-haul capabilities and even better reliability, but at a significant cost. Some find that copper terminations and cabling are more rugged and reliable than fiber. As always, refer to your card manufacturer's Web site for updated driver information.
- When using Gigabit Ethernet, ensure both ends of every link have enabled full flow control.

## 3.2 Linux utilities

Several utilities can be used to review and enable flow control on a Linux client, depending on the distribution you use.

### 3.2.1 mii-tool [-v]

This utility can be used to view and manipulate media-independent interface (MII) status (Figure 3-1). MII is used by fast Ethernet adapters to autonegotiate network link speed and duplex setting.

```
[root@itso ~]#
[root@itso ~]# mii-tool
eth1: negotiated 100baseTx-FD, link ok
[root@itso ~]#
[root@itso ~]# mii-tool -v
eth1: negotiated 100baseTx-FD, link ok
  product info: vendor 00:00:1a, model 54 rev 0
  basic mode:   autonegotiation enabled
  basic status: autonegotiation complete, link ok
  capabilities: 100baseTx-FD 100baseTx-HD 10baseT-FD 10baseT-HD
  advertising:  100baseTx-FD 100baseTx-HD 10baseT-FD 10baseT-HD
  link partner: 100baseTx-FD 100baseTx-HD 10baseT-FD 10baseT-HD
[root@itso ~]#
```

*Figure 3-1 An example of Linux utility mii*

### 3.2.2 ethtool [ethernet interface]

Ethtool is a Linux network driver diagnostic and tuning tool for Linux kernel version 2.4 or later. It obtains current network device settings and diagnostics details related to media, link status, driver version, PCI (or other) bus location, and more (Figure 3-2).

```
[root@itso ~]#
[root@itso ~]# ethtool eth1
Settings for eth1:
        Supported ports: [ TP MII ]
        Supported link modes:   10baseT/Half 10baseT/Full
                                100baseT/Half 100baseT/Full
        Supports auto-negotiation: Yes
        Advertised link modes:  10baseT/Half 10baseT/Full
                                100baseT/Half 100baseT/Full
        Advertised auto-negotiation: Yes
        Speed: 100Mb/s
        Duplex: Full
        Port: MII
        PHYAD: 30
        Transceiver: internal
        Auto-negotiation: on
        Current message level: 0x00000007 (7)
        Link detected: yes
[root@itso ~]#
```

*Figure 3-2 The Linux utility ethtool*

**Note:** Some switches, particularly mid-range switches, do not support flow control in both directions. Discuss support for full flow control with your switch vendor to ensure that your gigabit NIC and routers support it properly.

## 3.3 Jumbo frames

If you use Linux NFS clients and N series storage systems together on an un-routed network, consider using *jumbo frames* to improve the performance of your application (Figure 3-3). Be sure to consult your switch's command reference to make sure it is capable of handling jumbo frames in your environment.

```
itsotuc1>
itsotuc1> ifconfig e0a
e0a: flags=848043<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 192.168.3.254 netmask 0xffffff00 broadcast 192.168.3.255
        ether 00:a0:98:01:ff:c2 (auto-100tx-fd-up) flowcontrol full
itsotuc1>
itsotuc1> ifconfig e0a mtusize 8160
itsotuc1> Thu Dec 14 18:11:44 MST [itsotuc1: SB1250-Gigabit/e0a:info]: Eth
e0a: Link being reconfigured
Thu Dec 14 18:11:44 MST [itsotuc1: SB1250-Gigabit/e0a:info]: Ethernet e0a:
up

itsotuc1> ifconfig e0a
e0a: flags=848043<UP,BROADCAST,RUNNING,MULTICAST> mtu 8160
        inet 192.168.3.254 netmask 0xffffff00 broadcast 192.168.3.255
        ether 00:a0:98:01:ff:c2 (auto-100tx-fd-up) flowcontrol full
itsotuc1> Thu Dec 14 18:12:29 MST [itsotuc1: nbt.nbns.registrationComplete
: NBT: All CIFS name registrations have completed for the local server.
```

*Figure 3-3 Changing default MTU 1500 to use MTU 8160 jumbo frame option*

**Note:** It has been reported that an MTU setting higher than 8000 may degrade network throughput in some Linux environments. Even when using jumbo frames on more complex networks:

- Ensure that every link in the network between your client and server support them and have the support enabled. If NFS over TCP is working with jumbo frames, but NFS over UDP is not, that may be a sign that some part of your network does not support jumbo frames.
- Increasing the application's socket buffer size or increasing the /proc/sys/net/ipv4/tcp_*mem entry values, or both, may help.
- Reduce the MTU to 8960 or lower if you experience unexpected performance slowdowns when using maximum frame size 9000 bytes, and monitor the results.

The Linux NFS client and network layer are sensitive to network performance and reliability. After you have set your mount options as we recommend, you should get reasonable performance. If you do not and your workload is not already CPU-bound, you should begin looking at network conditions between your clients and servers.

If there is other network traffic or packet loss, write performance from a Linux client on NFS over UDP can drop rapidly, though on NFS over TCP, performance should remain reasonable. Read performance depends on the size and speed of the client's and the N series storage system's memory.

Features such as checksum offloading are beneficial to performance. You can use Ethernet bonding, or trunking, to improve the reliability or performance of your client.

Most network interface cards use auto negotiation to obtain the fastest settings allowed by the card and the switch port to which it attaches. Sometimes, PHY (Physical Layer) chipset incompatibilities may result in constant renegotiation or negotiating half-duplex or a slow speed. When diagnosing a network problem, be sure your Ethernet settings are as you expect before looking for other problems. To solve an auto negotiation problem, you may be inclined to hard code the settings you want, but avoid this because it only masks a deeper problem. Work with your switch and card vendors to resolve these problems.

Whether you use TCP or UDP, be sure you have a clean network. Ensure your network cards always negotiate the fastest settings, and that your NIC drivers are up to date. Increasing the size of the client's socket buffer (see Appendix A.1, "Special network settings" on page 57) is always a wise thing to do when using NFS over UDP.

## 3.4 Diagnosing performance problems with Linux NFS client

The client works best when the network does not drop any packets. The NFS and RPC clients also compete with applications for available CPU resources. These are the two main categories of client performance problems you may encounter.

Checking for network packet loss is the first area to look for problems. With NFS over UDP, a high retransmission count can indicate packet loss due to network or server problems. With NFS over TCP, the network layer on the client handles network packet loss, but server problems still show up as retransmissions. On some 2.4 kernels, TCP retransmissions are also a sign of large application writes on the client that have filled the RPC transport socket's output buffer.

```
itsotuc1> nfsstat -c

Server rpc:
TCP:
calls       badcalls    nullrecv    badlen      xdrcall
56          0           0           0           0

UDP:
calls       badcalls    nullrecv    badlen      xdrcall
0           0           0           0           0

Server nfs:
calls       badcalls
54          0

Server nfs V2: (0 calls)
null        getattr     setattr     root        lookup      readlink    read
0 0%        0 0%        0 0%        0 0%        0 0%        0 0%        0 0%
wrcache     write       create      remove      rename      link        symlink
0 0%        0 0%        0 0%        0 0%        0 0%        0 0%        0 0%
mkdir       rmdir       readdir     statfs
0 0%        0 0%        0 0%        0 0%

Read request stats (version 2)
0-511       512-1023    1K-2047     2K-4095     4K-8191     8K-16383    16K-3276
-65535   64K-131071
0           0           0           0           0           0           0
         0
Write request stats (version 2)
0-511       512-1023    1K-2047     2K-4095     4K-8191     8K-16383    16K-3276
-65535   64K-131071
0           0           0           0           0           0           0
         0


Server nfs V3: (54 calls)
null        getattr     setattr     lookup      access      readlink    read
21 39%      6 11%       0 0%        3 6%        3 6%        0 0%        0 0%
write       create      mkdir       symlink     mknod       remove      rmdir
0 0%        0 0%        0 0%        0 0%        0 0%        0 0%        0 0%
rename      link        readdir     readdir+    fsstat      fsinfo      pathconf
0 0%        0 0%        0 0%        1 2%        6 11%       14 26%      0 0%
commit
0 0%
```

*Figure 3-4 Example nfsstat -c command*

To see retransmissions, you can use **nfsstat –c** at a shell prompt (Figure 3-4). At the top of the output, the total number of RPCs the client has sent and the number of times the client had to retransmit an RPC are shown. The retransmit rate is determined by dividing the number of retransmissions by the total number of RPCs. If the rate exceeds a few tenths of a percent, network losses may be a problem for your performance.

NFS over TCP does not show up network problems as clearly as UDP and performs better in the face of packet loss. If your TCP mounts run faster than your UDP mounts, that is a sure sign that the network between your clients and your filer is dropping packets or is otherwise bandwidth-limited. Normally UDP is as fast as or slightly faster than TCP.

The client keeps network statistics that you can view with **netstat –s** at a shell prompt (Example 3-1). Look for high error counts in the IP, UDP, and TCP sections of this command's output. The same command also works on an N series storage system's console. Here, look for nonzero counts in the "fragments dropped after timeout" and "fragments dropped (dup or out of space)" fields in the IP section.

*Example 3-1 netstat -s*

```
[root@itsohelvio root]# netstat -s
Ip:
    4510550 total packets received
    1 with invalid headers
    0 forwarded
    165 incoming packets discarded
    4254173 incoming packets delivered
```

```
    4094715 requests sent out
Icmp:
    268034 ICMP messages received
    54325 input ICMP message failed.
    ICMP input histogram:
        destination unreachable: 253447
        echo requests: 44
        echo replies: 14543
    224274 ICMP messages sent
    0 ICMP messages failed
    ICMP output histogram:
        destination unreachable: 224230
        echo replies: 44
Tcp:
    287643 active connections openings
    103992 passive connection openings
    102306 failed connection attempts
    109303 connection resets received
    3 connections established
    2411927 segments received
    2606302 segments send out
    70164 segments retransmited
    0 bad segments received.
    4181 resets sent
Udp:
    1143580 packets received
    477 packets to unknown port received.
    0 packet receive errors
    1247032 packets sent
TcpExt:
    1 ICMP packets dropped because socket was locked
    ArpFilter: 0
    7 TCP sockets finished time wait in fast timer
    12 time wait sockets recycled by time stamp
    46009 delayed acks sent
    433290 packets directly queued to recvmsg prequeue.
    10785916 packets directly received from backlog
    18540949 packets directly received from prequeue
    219910 packets header predicted
    214916 packets header predicted and directly queued to user
    TCPPureAcks: 135098
    TCPHPAcks: 846310
    TCPRenoRecovery: 0
    TCPSackRecovery: 0
    TCPSACKReneging: 0
    TCPFACKReorder: 0
    TCPSACKReorder: 0
    TCPRenoReorder: 0
    TCPTSReorder: 0
    TCPFullUndo: 0
    TCPPartialUndo: 0
    TCPDSACKUndo: 0
    TCPLossUndo: 940
    TCPLoss: 0
    TCPLostRetransmit: 0
```

```
TCPRenoFailures: 0
TCPSackFailures: 0
TCPLossFailures: 0
TCPFastRetrans: 0
TCPForwardRetrans: 0
TCPSlowStartRetrans: 0
TCPTimeouts: 70164
TCPRenoRecoveryFail: 0
TCPSackRecoveryFail: 0
TCPSchedulerFailed: 8
TCPRcvCollapsed: 0
TCPDSACKOldSent: 0
TCPDSACKOfoSent: 0
TCPDSACKRecv: 0
TCPDSACKOfoRecv: 0
TCPAbortOnSyn: 0
TCPAbortOnData: 2866
TCPAbortOnClose: 106580
TCPAbortOnMemory: 0
TCPAbortOnTimeout: 0
TCPAbortOnLinger: 0
TCPAbortFailed: 0
TCPMemoryPressures: 0
```

There are a few basic sources of packet loss:

- If the end-to-end connection between your clients and servers contains links of different speeds, packet loss occurs at the point where the two speeds meet. UDP does not have any flow control built in to slow the server's transmission rate, but TCP does; thus, it provides reasonable performance through a link speed change.
- Another source of packet loss is small packet buffers on switches. If either the client or server bursts a large number of packets, the switch may buffer them before sending them on. If the switch buffer overflows, the packets are lost. It is also possible that a switch can overrun a client's NIC in a similar fashion. This becomes a greater possibility for large UDP datagrams because switches and NICs tend to burst an entire IP packet (all of its fragments) at once.
- The client's IP layer will drop UDP datagrams if the client's send or receive socket buffer runs out of space for any reason. The client's RPC client allocates a socket for each mount. By default, these sockets use 64KB input and output buffers, which is too small on systems that use large rsize or wsize or generate a large number of NFS operations in a short period. To increase the size of these buffers, refer to Appendix A.1, "Special network settings" on page 57 in this document. There is no harm in doing this on all your Linux clients unless they have less than 16MB of RAM.

### 3.4.1 Troubleshooting utilities

If you have resolved these issues and still have poor performance, you can attempt end-to-end performance testing between one of your clients and a similar system on the server's LAN using tools such as the following:

- ttcp

  Test TCP (TTCP) is a command line sockets-based benchmarking tool for measuring TCP and UDP performance between two systems. It was originally developed for the BSD

operating system starting in 1984. There are 2 programs, one of them (ttcp1.c) is the transmitter and the other (ttcp2.c) the receiver. The receiver must be installed before the loading is started. After the codes are compiled and executed, it will create two executable files for you: tcp1 and tcp2.

To run tcp1(as transmitter), use the following:

```
tcp1 [-options] host
```

To run tcp2 (as receiver), use the following:

```
./tcp2 [-dv] [-b bytes] [-u] [-p port] [-w bytes]
```

- iPerf

  iPerf was developed by NLANR/DAST as a tool for measuring maximum TCP and UDP bandwidth performance. Basically, it reports information such as bandwidth, transfer statistics, datagram loss, and so forth. (Example 3-3, Figure 3-5).

  This exposes problems that occur in the network outside of the NFS protocol. When running tests such as iPerf, select UDP tests because these directly expose network problems. If your network is full duplex, run iPerf tests in both directions concurrently to ensure your network is capable of handling a full load of traffic in both directions simultaneously.

  If you find some client operations work normally while others cause the client to stop responding, try reducing your rsize and wsize to 1024 to see if there are fragmentation problems preventing large operations from succeeding.

*Example 3-2 Sample output from the server*

```
node2> iperf -s -w 130k
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size:  130 KByte
------------------------------------------------------------
[  4] local <IP Addr node 2> port 5001 connected with <IP Addr node 1> port 2530
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-10.1 sec  19.7 MBytes  15.7 Mbits/sec
```

*Example 3-3 Sample output from the Client*

```
node1> iperf -c node2 -w 130k
------------------------------------------------------------
Client connecting to node2, TCP port 5001
TCP window size:  129 KByte (WARNING: requested  130 KByte)
------------------------------------------------------------
[  3] local <IP Addr node1> port 2530 connected with <IP Addr node2> port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  19.7 MBytes  15.8 Mbits/sec
```

On the client, you can run the network snooping tools such as:

- tcpdump / tcpslice
  - tcpdump prints out the headers of packets on a network interface that match the boolean expression (Figure 3-5).
  - tcpslice is a program for extracting portions of packet-trace output files generated using `tcpdump` with `-w` flag.

```
itso ~]# tcpdump host 192.168.3.246 -v
p: listening on eth1, link-type EN10MB (Ethernet), capture size 96 bytes
30.058289 IP (tos 0x0, ttl 128, id 16055, offset 0, flags [DF], proto 6, l
 40) 192.168.3.242.3751 > itso.telnet: . [tcp sum ok] ack 377244469 win 64

30.060324 IP (tos 0x0, ttl  64, id 13999, offset 0, flags [DF], proto 6, l
 120) itso.telnet > 192.168.3.242.3751: P 1:81(80) ack 0 win 5840
30.059447 IP (tos 0x0, ttl  64, id 34384, offset 0, flags [DF], proto 17,
: 72) itso.1031 > 192.168.3.242.domain:  53809+ PTR? 242.3.168.192.in-addr
 (44)
30.060034 IP (tos 0x0, ttl 128, id 16057, offset 0, flags [none], proto 17
th: 149) 192.168.3.242.domain > itso.1031:  53809 NXDomain 0/1/0 (121)
30.276948 IP (tos 0x0, ttl 128, id 16059, offset 0, flags [DF], proto 6, l
 40) 192.168.3.242.3751 > itso.telnet: . [tcp sum ok] ack 81 win 64344
30.276987 IP (tos 0x0, ttl  64, id 14001, offset 0, flags [DF], proto 6, l
 689) itso.telnet > 192.168.3.242.3751: P 81:730(649) ack 0 win 5840
30.495852 IP (tos 0x0, ttl 128, id 16061, offset 0, flags [DF], proto 6, l
 40) 192.168.3.242.3751 > itso.telnet: . [tcp sum ok] ack 730 win 65535
30.495894 IP (tos 0x0, ttl  64, id 14003, offset 0, flags [DF], proto 6, l
 354) itso.telnet > 192.168.3.242.3751: P 730:1044(314) ack 0 win 5840
30.714570 IP (tos 0x0, ttl 128, id 16063, offset 0, flags [DF], proto 6, l
 40) 192.168.3.242.3751 > itso.telnet: . [tcp sum ok] ack 1044 win 65221
30.714599 IP (tos 0x0, ttl  64, id 14005, offset 0, flags [DF], proto 6, l
 357) itso.telnet > 192.168.3.242.3751: P 1044:1361(317) ack 0 win 5840
30.933257 IP (tos 0x0, ttl 128, id 16065, offset 0, flags [DF], proto 6, l
 40) 192.168.3.242.3751 > itso.telnet: . [tcp sum ok] ack 1361 win 64904
30.933273 IP (tos 0x0, ttl  64, id 14007, offset 0, flags [DF], proto 6, l
 359) itso.telnet > 192.168.3.242.3751: P 1361:1680(319) ack 0 win 5840
31.151994 IP (tos 0x0, ttl 128, id 16067, offset 0, flags [DF], proto 6, l
 40) 192.168.3.242.3751 > itso.telnet: . [tcp sum ok] ack 1680 win 64585
31.152010 IP (tos 0x0, ttl  64, id 14009, offset 0, flags [DF], proto 6, l
 359) itso.telnet > 192.168.3.242.3751: P 1680:1999(319) ack 0 win 5840
31.370694 IP (tos 0x0, ttl 128, id 16069, offset 0, flags [DF], proto 6, l
 40) 192.168.3.242.3751 > itso.telnet: . [tcp sum ok] ack 1999 win 64266
31.370710 IP (tos 0x0, ttl  64, id 14011, offset 0, flags [DF], proto 6, l
 359) itso.telnet > 192.168.3.242.3751: P 1999:2318(319) ack 0 win 5840
31.589405 IP (tos 0x0, ttl 128, id 16071, offset 0, flags [DF], proto 6, l
 40) 192.168.3.242.3751 > itso.telnet: . [tcp sum ok] ack 2318 win 65535
```

*Figure 3-5 An example of Linux utility tcpdump*

- ethereal

  ethereal is a network protocol analyzer for UNIX and Windows®. It has an extensive list of features to capture network traffic for troubleshooting or performance monitoring (Figure 3-6).

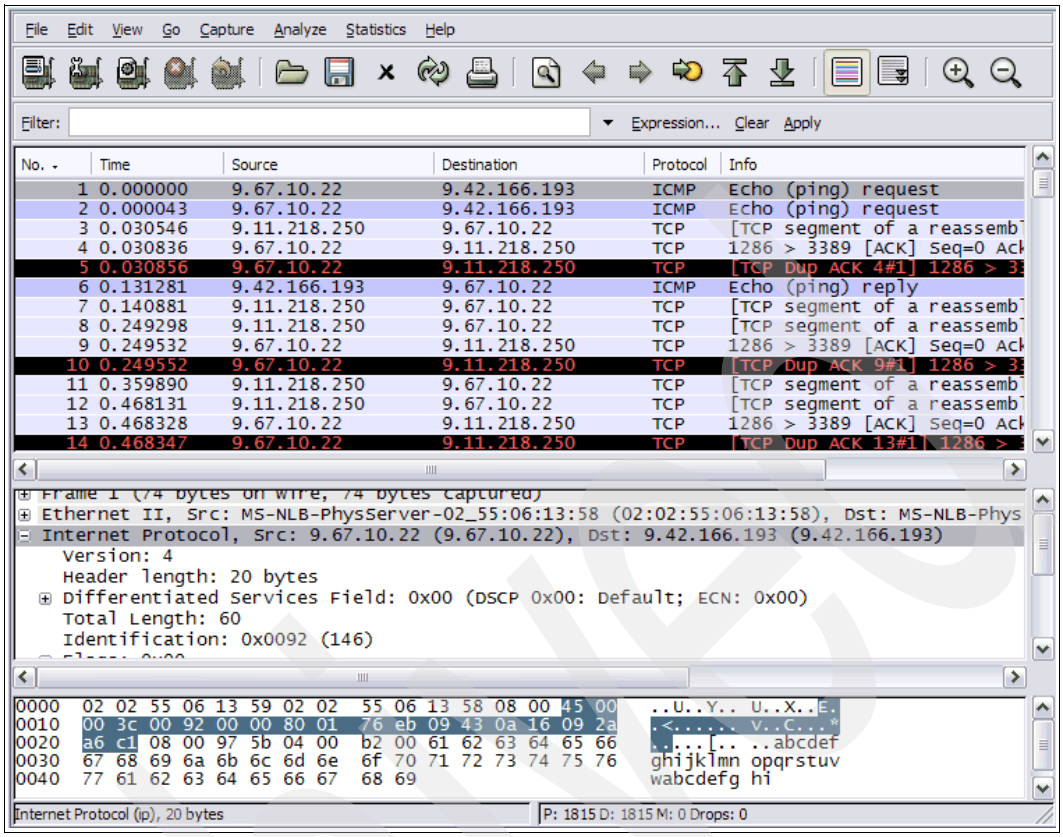


*Figure 3-6 Network protocol analyzer etheral*

On the N series storage system, you can run Data ONTAP network trace tools such as:

- `pktt`

  Pktt generates trace files in tcpdump format that you can analyze later on a client (Figure 3-7). To run pktt, use the following utilities on your N series storage system (Example 3-4).

*Example 3-4 Sample Data ONTAP pktt utilities*

```
pktt start <interface>|all [-d dir] [-s size] [-m pklen] [-b bsize] [-i ip@..]
[-v]
pktt pause <interface>|all
pktt dump <interface>|all [-d dir]
pktt stop <interface>|all
pktt status [<interface>|all] [-v]
```

```
itsotuc1>
itsotuc1> pktt start e0a -d /vol/vol0 -i 192.168.3.242
e0a: started packet trace
itsotuc1>
itsotuc1> pktt status
e0a: Packet tracing enabled; packets truncated at 1514 bytes.
e0a: Trace buffer utilization = 0% of 1048320 bytes, 61 packets
e0a: 0 bytes written to file /vol/vol0/e0a_20061216_123947.trc
e0a: Currently tracing to file /vol/vol0/e0a_20061216_123947.trc
e0a: 61 packets seen; 0 packets dropped; 3940 total bytes seen

e0b: packet tracing not enabled

lo: packet tracing not enabled

itsotuc1>
itsotuc1> pktt stop e0a
e0a: Tracing stopped and packet trace buffers released.
itsotuc1> Sat Dec 16 12:40:10 MST [itsotuc1: rc:info]: pktt: 105 packets seen,
 dropped, 9214 bytes written to /vol/vol0/e0a_20061216_123947.trc
```

*Figure 3-7 Example of Data ONTAP iptrace utility pktt*

These tools provide the last word in what is really happening on your network between your clients and N series storage systems. You may need to run both tcpdump on a client and pktt on your N series storage system at the same time and compare the traces to determine where the problem lies.

You must explicitly specify several options to collect clean network traces with tcpdump. Be sure the snaplen option (`-s`) is set large enough to capture all the interesting bytes in each packet, but small enough that tcpdump is not overwhelmed with incoming traffic. If tcpdump is overwhelmed, it drops incoming packets, making the network trace incomplete. The default value is 96 bytes, which is too short to capture all the RPC and NFS headers in each packet. Usually a value of 256 bytes is a good compromise for UDP, but you can set it to zero if you need to see all the data in each packet. Snooping TCP packets requires a zero snaplen because TCP can place several RPC requests in a single network packet. If snaplen is short, the trace will miss RPCs that are contained near the end of long packets.

In addition, always use filtering to capture just the traffic between the client and the server. Again, this reduces the likelihood that tcpdump or your local filesystem will be overwhelmed by incoming traffic and makes later analysis easier. You can collect traffic to or from your client using the hostname filter. Several other tcpdump options allow you to collect traffic destined for one or more hosts at a time; read the manual to find out more.

An automounter can cause a lot of network chatter, so it is best to disable the automounter on your client and set up your mounts by hand before taking a network trace (Example 3-5).

*Example 3-5 Disabling automounter on the client*

```
cd /etc
rm auto_*
cd /etc/rc2.d
mv S74autofs OFF_S74autofs
```

To find out if your application is competing for CPU resources with the NFS client, RPC client, or network layer on your client system, you can use the `top` program. If you see the `rpciod` process at the top of the listing, you know that NFS operations are dominating the CPU on your system. In addition, if you see system CPU percentage increase significantly when your application accesses NFS data, this also can indicate a CPU shortage. In many cases, adding more CPUs or faster CPUs helps. Switching to UDP may also be a choice for reducing CPU load if an uncongested high-speed network connects your clients and N series storage systems. As the Linux NFS client and networking layer improve over time, they will become more CPU-efficient and include features such as TCP offload that reduce the amount of CPU required to handle large amounts of data.

There are certain cases where processes accessing NFS filesystems may hang. This is most often due to a network partition or server outage. Today's client implementation is robust enough to recover in most cases. Occasionally a client fails because of high load or some other problem. Unfortunately, little can be done in these cases other than rebooting the client and reporting the problem.

4

# Error messages

This chapter describes several common error messages you might encounter.

## 4.1 Error messages in the kernel log

There are two messages that you might encounter frequently in the kernel log (this is located in /var/log/messages on Linux systems):

- "`server not responding`"

  This message occurs after the client retransmits several times without any response from a server. If you know the server is up, this can indicate the server is sluggish or there are network problems. If you know the server is down, this indicates the client is waiting for outstanding operations to complete on that server, and it is likely there are programs waiting for the server to respond.

- "`can't get request slot`"

  This message indicates that the RPC client is queuing messages and cannot send them. This is usually due to network problems such as a bad cable, incorrectly set duplex or flow control options, or an overloaded switch. It may appear as if your client is stuck at this point, but you should always wait at least 15 minutes for network and RPC client timeouts to recover before trying harsher remedies such as rebooting your client or filer.

## 4.2 Oops

If a program encounters an unrecoverable situation in user space, it stops running and dumps core. If the same process encounters an unrecoverable situation within the Linux kernel, the kernel attempts to isolate the process, stop it, and report the event in the kernel log. This is called an "*oops*."

Many times a failing process holds system resources that are not released during an oops. The kernel can run for a few more moments, but generally other processes deadlock or themselves terminate with an oops when they attempt to allocate resources that were held by the original failing process.

If you are lucky, the kernel log contains useful information after the system recovers. Red Hat kernels automatically translate the kernel log output into symbolic information that means something to kernel developers. If the oops record contains only hexadecimal addresses, try using the ksymoops tool in the kernel source tree to decode the addresses.

Sometimes using a serial console setup can help capture output that a failing kernel cannot write into its log. You will need to do the following:

- Use a crossover cable (also sometimes called a file transfer or null modem cable) to connect your client to another system's serial console via its COM1 or COM2 port.
- On the receiving system, use a tool such as the minicom program to access the serial port. Serial console support is built into kernels distributed by Red Hat, but be sure to enable the serial console option in the kernel you build yourself.
- Finally, you should set appropriate boot command line options (instructions provided in the Documentation directory of the Linux kernel source tree) to finish enabling the serial console.
- Optionally, you can also update /etc/inittab to start a `mingetty` process on your serial console if you'd like to log in there (Example 4-1).

```
[root@itso etc]# cat /etc/init.tab
cat: /etc/init.tab: No such file or directory
[root@itso etc]# cd
[root@itso ~]#
[root@itso ~]# cd /etc/inittab
-bash: cd: /etc/inittab: Not a directory
[root@itso ~]# cat /etc/inittab
#
# inittab       This file describes how the INIT process should set up
#               the system in a certain run-level.
#
# Author:       Miquel van Smoorenburg, <miquels@drinkel.nl.mugnet.org>
#               Modified for RHS Linux by Marc Ewing and Donnie Barnes
#

# Default runlevel. The runlevels used by RHS are:
#   0 - halt (Do NOT set initdefault to this)
#   1 - Single user mode
#   2 - Multiuser, without NFS (The same as 3, if you do not have networki
#   3 - Full multiuser mode
#   4 - unused
#   5 - X11
#   6 - reboot (Do NOT set initdefault to this)
#
id:5:initdefault:

# System initialization.
si::sysinit:/etc/rc.d/rc.sysinit

l0:0:wait:/etc/rc.d/rc 0
l1:1:wait:/etc/rc.d/rc 1
l2:2:wait:/etc/rc.d/rc 2
l3:3:wait:/etc/rc.d/rc 3
l4:4:wait:/etc/rc.d/rc 4
l5:5:wait:/etc/rc.d/rc 5
l6:6:wait:/etc/rc.d/rc 6

# Trap CTRL-ALT-DELETE
ca::ctrlaltdel:/sbin/shutdown -t3 -r now

# When our UPS tells us power has failed, assume we have a few minutes
# of power left.  Schedule a shutdown for 2 minutes from now.
# This does, of course, assume you have powerd installed and your
# UPS connected and working correctly.
pf::powerfail:/sbin/shutdown -f -h +2 "Power Failure; System Shutting Down

# If power was restored before the shutdown kicked in, cancel it.
pr:12345:powerokwait:/sbin/shutdown -c "Power Restored; Shutdown Cancelled


# Run gettys in standard runlevels
1:2345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
```

*Figure 4-1 Start a `mingetty` process in /etc/inittab file*

## 4.3 Getting help

Most Linux NFS client performance problems are due to lack of CPU or memory on the client, incorrect mount options, or packet losses on the network between the client and servers. If you have set up your client correctly and your network is clean, but you still suffer from performance or reliability problems, you should contact experts to help you proceed further.

Expert help is available on the Web at:

`http://nfs.sourceforge.net`

Here you can find a Linux NFS Frequently Asked Questions (FAQ) list, as well as several how-to documents. There is also a mailing list specifically for helping administrators get the best from Linux NFS clients and servers.

If you find there are missing features or performance or reliability problems, we encourage you to participate in the community development process. Unlike proprietary operating systems, new features appear in Linux only when users implement them. Problems are fixed

when users are diligent about reporting them and following up to see that they are really fixed. If you have ever complained about the Linux NFS client, here is your opportunity to do something about it.

When you have found a problem with the Linux NFS client, you can report it to your Linux distributor. Red Hat, for instance, supports an online bug database based on bugzilla. You can access Red Hat's bugzilla instance at:

http://bugzilla.redhat.com/

5

# Support for NFS features

This section covers auxiliary services you may need to support advanced NFS features.

## 5.1 Telling time

The clock on your Linux clients must remain synchronized with your N series storage systems to avoid problems such as authentication failures or incomplete software builds. Usually you set up a network time service such as NTP and configure your N series storage systems and clients to update their time using this service. After you have properly configured a network time service, you can find more information on enabling NTP on your N series storage systems in the *Data ONTAP System Administrator's Guide* at:

`http://www.ibm.com/support/us`

Linux distributions usually come with a prebuilt network time protocol daemon. If your distribution does not have an NTP daemon, you can build and install one yourself by downloading the latest *ntpd* package from the Internet (see Appendix A.1, "Special network settings" on page 57).

There is little documentation available for the pre-installed NTP daemon on Linux. To enable NTP on your clients, be sure the ntpd startup script runs when your client boots. (Look in /etc/rc.d or /etc/init.d; the exact location varies, depending on your distribution. For Red Hat systems, you can use `chkconfig –level 35 ntpd on`). You must add the network time server's IP address to /etc/ntp/step-tickers and /etc/ntp.conf.

If you find that the time protocol daemon is having some difficulty maintaining synchronization with your time servers, you may need to create a new drift file. Make sure your client's /etc/ntp directory and its contents are permitted to the ntp user and group to allow the daemon to update the drift file, and disable authentication and restriction commands in /etc/ntp.conf until you are sure everything is working correctly.

As root, shut down the time daemon and delete the drift file (usually /etc/ntp/drift). Now restart the time daemon again. After about 90 minutes, it will write a new drift file into /etc/ntp/drift. Your client system should keep better time after that.

Always keep the date, time, and time zone on your filer and clients synchronized. Not only will you ensure that any time-based caching on your clients works correctly, but it will also make debugging easier by aligning time stamps in client logs and on client network trace events with the N series storage system's message log and pktt traces.

## 5.2 Security

Today, most versions of the Linux NFS client support only two types of authentication:

- AUTH_NULL
- AUTH_UNIX

Linux distributions based on 2.6 kernels support Kerberos 5, just as Solaris™ does today, via RPCSEC GSS. Later versions of the NFS protocol (for example, NFSv4) support a wide variety of authentication and security models, including Kerberos, and a form of public key authentication called SPKM. In this section, we cover some security basics, and then describe how to configure the Linux NFS client to use Kerberos.

To maintain the overall security of your Linux clients, be sure to check for and install the latest security updates from your distributor. For more details on Linux NFS FAQ, see Appendix A.1, "Special network settings" on page 57.

When a host wants to send UDP packets that are larger than the network's maximum transfer unit, or MTU, it must fragment them. Linux divides large UDP packets into MTU-sized IP fragments and sends them to the receiving host in reverse order; that is, the fragment that contains the bytes with the highest offset are sent first. Because Linux sends IP fragments in reverse order, its NFS client may not interoperate with some firewalls. Certain modern firewalls examine the first fragment in a packet to determine whether to pass the rest of the fragments. If the fragments arrive in reverse order, the firewall discards the whole packet. A possible workaround is to use only NFS over TCP when crossing such firewalls. If you find some client operations work normally while others cause the client to stop responding, try reducing your rsize and wsize to 1024 to see if there are fragmentation problems preventing large RPC requests from succeeding.

Firewall configuration can also block auxiliary ports that the NFS protocol requires to operate. For example, traffic may be able to pass on the main NFS port numbers, but if a firewall blocks the mount protocol or the lock manager or port manager ports, NFS cannot work. This applies to standalone router/firewall systems as well as local firewall applications such as tcpwrapper, ipchains, or iptables that might run on the client system itself. Be sure to check whether there are any rules in /etc/hosts.deny that might prevent communications between your client and server.

The Linux NFS client in 2.6 kernels supports NFS with Kerberos. The implementation is not yet mature, so there are some limitations you should be aware of as you consider deploying it. First, NFS with Kerberos is supported for all three versions of NFS. However, because NFS version 4 has combined all the auxiliary protocols (NLM, NSM, mountd, and ACL) into one protocol, it is not exposed to traditional attacks on Kerberized NFS. Finally, NFS with Kerberos is supported only with Kerberos 5; you cannot use NFS with Kerberos in a Kerberos 4 realm. However, NFS with Kerberos works with either Active Directory® Kerberos or with MIT Kerberos hosted on a UNIX system.

The Linux NFS client has some trouble with NFS over UDP with Kerberos. We recommend that if you want to use NFS with Kerberos, you should use NFS over TCP. As mentioned previously, the NFS client in RHEL 4.0 and 2.6-based Fedora Core distributions supports only Kerberos authentication and Kerberos authentication with request integrity checking. And, support for SECINFO and WRONGSEC is still missing from the 2.6 NFS client, so there is still no support for crossing NFSv4 mounts that use different security flavors.

The first step for configuring your client is to ensure the local UID mapper is configured and running. See 2.3, "Mounting with NFS version 4" on page 23 for more details and instructions. An overview describing how to configure Kerberos on your Linux client for use with NFS is available here:

`http://www.citi.umich.edu/projects/nfsv4/`

- To begin configuring Kerberos itself, make sure your /etc/krb5.conf reflects the proper configuration of your local Kerberos realm. You can use **`authconfig`** on your Red Hat clients to provide information about your local Kerberos realm. If you are not sure what this means, contact your local Kerberos administrator.
- The next step is to acquire a host key for your client, stored in /etc/krb5.keytab. This file (by convention) stores Kerberos keys for use on your client. A normal user types in a password which is transformed into a key. For a service or daemon, the key is stored in a file without the need to enter a password. A keytab entry allows a service to verify a client's identification. It can also be used by a daemon to authenticate to another Kerberized service. Your local Kerberos administrator can help you with this since Linux uses the same standard Kerberos commands found on most UNIX systems. The form of the key is:

  'nfs/<client hostname>@<KERBEROS REALM>'

- Your N series storage system will also need a keytab and similar configuration, and you need to add an appropriate set of `sec=` export rules in your filer's /etc/exports file. Instructions describing how to do this can be found in recent releases of Data ONTAP documentation at:

  `http://www.ibm.com/support/us`

- Now you are ready to start the client's GSS daemon. On some distributions, you may need to add the string `SECURE_NFS=YES` to the file /etc/sysconfig/nfs. Then, start the daemon with this command:

  ```
  /etc/init.d/rpcgssd start
  ```

  Some distributions use the **`chkconfig`** command to cause these scripts to run automatically when your client boots.

- To mount using NFS with Kerberos, simply add the `sec=krb5` mount option to your mount command line or in /etc/fstab. Kerberos integrity checking is enabled by using `sec=krb5i` instead of `sec=krb5`. This will require users to acquire Kerberos credentials before they can access files in NFS filesystems mounted with Kerberos.

  There are two typical ways in which users acquire these credentials. Users can use the **`kinit`** command after logging in. Or, you can configure your client to use the Kerberos PAM library to acquire the credentials automatically when users log in. Consult your distribution's documentation for more information on how to set up Kerberos and PAM security on your Linux client.

## 5.3 Network lock manager

The NFS version 2 and 3 protocols use separate side-band protocols to manage file locking. On Linux 2.4 kernels, the *`lockd`* daemon manages file locks using the NLM (Network Lock Manager) protocol, and the *`rpc.statd`* program manages lock recovery using the NSM (Network Status Monitor) protocol to report server and client reboots. The lockd daemon runs in the kernel and is started automatically when the kernel starts up at boot time. The rpc.statd program is a user-level process that is started during system initialization from an init script. If rpc.statd is not able to contact servers when the client starts up, stale locks will remain on the servers that can interfere with the normal operation of applications.

```
[root@itso mnt]# rpcinfo -p 192.168.3.254
   program vers proto   port
    100024    1   tcp   4047  status
    100024    1   udp   4047  status
    100011    1   udp   4049  rquotad
    100021    4   tcp   4045  nlockmgr
    100021    3   tcp   4045  nlockmgr
    100021    1   tcp   4045  nlockmgr
    100021    4   udp   4045  nlockmgr
    100021    3   udp   4045  nlockmgr
    100021    1   udp   4045  nlockmgr
    100005    3   tcp   4046  mountd
    100005    2   tcp   4046  mountd
    100005    1   tcp   4046  mountd
    100005    3   udp   4046  mountd
    100005    2   udp   4046  mountd
    100005    1   udp   4046  mountd
    100003    3   tcp   2049  nfs
    100003    2   tcp   2049  nfs
    100003    3   udp   2049  nfs
    100003    2   udp   2049  nfs
    100000    2   tcp    111  portmapper
    100000    2   udp    111  portmapper
```

*Figure 5-1 An example of rpcinfo command*

The **`rpcinfo`** command on Linux can help determine whether these services have started and are available (Figure 5-1). If rpc.statd is not running, use the chkconfig program to check that

its init script (which is usually /etc/init.d/nfslock) is enabled to run during system bootup. If the client host's network stack is not fully initialized when rpc.statd runs during system startup, rpc.statd may not send a reboot notification to all servers. Some reasons network stack initialization can be delayed are slow NIC devices, slow DHCP service, or CPU-intensive programs running during system startup. Network problems external to the client host may also cause these symptoms.

Because status monitoring requires bidirectional communication between server and client, some firewall configurations can prevent lock recovery from working. Firewalls may also significantly restrict communication between a client's lock manager and a server. Network traces captured on the client and server at the same time usually reveal a networking or firewall misconfiguration.

Your client's nodename determines how a filer recognizes file lock owners. You can easily find out what your client's nodename is using the **`uname –n`** (Figure 5-2) or **`hostname`** command. (A system's nodename is set on Red Hat systems during boot using the HOSTNAME value set in /etc/sysconfig/network.) The rpc.statd daemon determines which name to use by calling `gethostbyname(3)`, or you can specify it explicitly when starting rpc.statd using the `-n` option.

```
[root@itso ~]#
[root@itso ~]# uname --a
Linux itso 2.6.9-34.ELsmp #1 SMP Fri Feb 24 16:54:53 EST 2006 i686 i686 i3
/Linux
[root@itso ~]# uname -r
2.6.9-34.ELsmp
[root@itso ~]# uname -n
itso
[root@itso ~]#
```

*Figure 5-2 An example of Linux uname command*

If the client's nodename is fully qualified (that is, it contains the hostname and the domain name spelled out), then rpc.statd must also use a fully qualified name. Likewise, if the nodename is unqualified, then rpc.statd must use an unqualified name. If the two values do not match, lock recovery will not work. Be sure the result of `gethostbyname(3)` matches the output of `uname –n` by adjusting your client's nodename in /etc/hosts, DNS, or your NIS databases.

Similarly, you should account for client hostname clashes in different sub domains by ensuring that you always use a fully qualified domain name when setting up a client's nodename during installation. With multi-homed hosts and aliased hostnames, you can use rpc.statd's `-n` option to set unique hostnames for each interface. The easiest approach is to use each client's fully qualified domain name as its nodename.

When working in high availability database environments, test all worst-case scenarios (such as server crash, client crash, application crash, network partition, and so on) to ensure lock recovery is functioning correctly before you deploy your database in a production environment. Ideally, you should examine network traces and the kernel log before, during, and after the locking/disaster/locking recovery events.

The filesystem containing /var/lib/nfs must be persistent across client reboots. This directory is where the rpc.statd program stores information about servers that are holding locks for the local NFS client. A tmpfs filesystem, for instance, is not sufficient; the server will fail to be notified that it must release any POSIX locks it might think your client is holding if it fails to shut down cleanly. That can cause a deadlock the next time you try to access a file that was locked before the client restarted.

Locking files in NFS can affect the performance of your application. The NFS client assumes that if an application locks and unlocks a file, it wishes to share that file's data among

cooperating applications running on multiple clients. When an application locks a file, the NFS client purges any data it has already cached for the file, forcing any read operation after the lock to go back to the server. When an application unlocks a file, the NFS client flushes any writes that may have occurred while the file was locked. In this way, the client greatly increases the probability that locking applications can see all previous changes to the file.

However, this increased data cache coherency comes at the cost of decreased performance. In some cases, all of the processes that share a file reside on the same client; thus aggressive cache purging and flushing unnecessarily hamper the performance of the application. Solaris allows administrators to disable the extra cache purging and flushing that occur when applications lock and unlock files with the `llock` mount option. Note well that this is not the same as the `nolock` mount option in Linux. The `nolock` mount option disables NLM calls by the client, but the client continues to use aggressive cache purging and flushing. Essentially this is the opposite of what Solaris does when `llock` is in effect.

## 5.4 Using the Linux automounter

Because Linux minor device numbers have only eight bits, a single client cannot mount more than 250 or so NFS filesystems. The major number for NFS mounts is the same as for other filesystems that do not associate a local disk with a mount point. These are known as *anonymous* filesystems. Because the NFS client shares the minor number range with other anonymous filesystems, the maximum number of mounted NFS filesystems can be even less than 250. In later releases of Linux, more anonymous device numbers are available, thus the limit is somewhat higher.

The preferred mechanism to work around this problem is to use an automounter. This also helps performance problems that occur when mounting very large root-level directories. There are two Linux automounters available: AMD and automounter. The autofs filesystem is required by both and comes built into modern Linux distributions. For more information on Linux automounters see Appendix A.1, “Special network settings” on page 57.

A known problem with the automounter in Linux is that it polls NFS servers on every port before actually completing each mount to be sure the mount request won’t hang. This can result in significant delays before an automounted filesystem becomes available. If your applications hang briefly when they transition into an automounted filesystem, make sure your network is clean and that the automounter is not using TCP to probe the N series storage system’s portmapper. Ensure your infrastructure services, such as DNS, respond quickly and with consistent results. Also consider upgrading your N series storage system to the latest release of Data ONTAP.

An automounter can cause a lot of network chatter, so it is best to disable the automounter on your client and set up static mounts before taking a network trace (Example 3-5 on page 43). Automounters depend on the availability of several network infrastructure services. If any of these services is not reliable or performs poorly, it can adversely affect the performance and availability of your NFS clients. When diagnosing an NFS client problem, triple-check your automounter configuration first. It is often wise to disable the automounter before drilling into client problem diagnosis.

The Linux automounter is a single process, and handles a single mount request at a time. If one such request becomes stuck, the automounter will no longer respond, causing applications to hang while waiting to enter a filesystem that has yet to be mounted. If you find hanging applications on a client that is managed with the automounter, be sure to check that the automounter is alive and is responding to requests.

Make sure mount operations do not occur during the small window when a fresh version of a SnapMirrored replica is brought online. If the automounter attempts to mount a volume during this brief window, it will fail, but a mount request moments later will succeed. There is no workaround for this problem by adjusting your automounter configuration, but upgrading to the latest version of Data ONTAP should resolve the issue.

Using an automounter is not recommended for production servers that may need immediate access to files after long periods of inactivity. Oracle, for example, may need immediate access to its archive files every so often, but an automounter may unmount the archive filesystem due to inactivity.

## 5.5 Network booting your Linux NFS clients

Intel systems can support network booting using DHCP, BOOTP, and TFTP. The Intel standard for supporting network booting is called a *preexecution* environment, or PXE. Usually this requires network interface hardware that contains a special PROM module that controls the network boot process. Network booting is especially helpful for managing clusters, blade servers, or a large number of workstations that are similarly configured.

Generally, Linux is loaded via a secondary loader such as grub, lilo, or syslinux. The secondary loader of choice for network booting is *pxelinux*, which comes in the *syslinux* distribution. See the appendix for information on how to obtain and install the syslinux distribution.

Data ONTAP supports pxelinux, allowing Linux to boot over a network. To enable TFTP access to your filer, refer to *Data ONTAP System Administrator's Guide* at:

`http://www.ibm.com/support/us`

You must ensure that your client hardware supports network booting. Both the client's mainboard and network interface card must have support for network booting built in. Usually you can tell whether network booting is supported by reviewing the BIOS settings on your client or by consulting the mainboard manual for information on how to set up network booting on your client hardware. The specific settings vary from manufacturer to manufacturer.

You must configure a DHCP server on the same LAN as your clients. A DHCP server provides unique network configuration information for each host on a commonly administered network. For network booting, the DHCP server also instructs each client where to find that client's boot image. You can find instructions for configuring your DHCP server to support network booting included with the pxelinux distribution. The specific instructions for setting up a DHCP server vary from vendor to vendor.

If you intend to share a common root filesystem among multiple Linux clients, you must create the root filesystem on your N series storage system using NFSv2. This is because of how N series storage systems interpret major and minor device numbers and because of differences between how NFSv2 and NFSv3 Linux clients transmit these numbers. Linux clients, unless told otherwise, attempt to mount their root filesystems with NFSv2. If the filesystem was created using NFSv3, the major and minor numbers will appear incorrect when mounted with NFSv2. Kernels use these numbers to match the correct device driver to device special files (files that represent character and block devices). If the numbers are mismatched, the Linux kernel will not be able to find its console or root filesystem, thus it cannot boot.

When setting up multiple clients with NFS root filesystems, common practice is to maintain a separate root filesystem for each client and mount each with `ro,nolock` options. Sharing a root filesystem among clients is not recommended.

Note that /var/lib/nfs must be persistent across reboots and unique for each client. A tmpfs filesystem per client, for instance, is not sufficient; the server will fail to be notified that it must release any POSIX locks it might think your client is holding if it fails to shut down cleanly. That can cause a deadlock the next time you try to access a file locked before the client restarted. If each client mounts a private /var/lib/nfs directory via NFS, it must be mounted using the `nolock` mount option, and before `rpc.statd` and `lockd` have started on the client.

For more information on Linux cluster computing, search the Internet for Beowulf or OpenMosix,

Make sure to check with your Linux distributor for any errata on a regular basis to maintain a secure system. Be sure your N series storage systems and clients agree on what time it is. If your client has trouble seeing the filer, look for packet filtering on your client or switches. Make each client's nodename its fully qualified domain name.

## 5.6 Summary

- When setting up a Linux NFS client, you should try to get the latest kernel supported by your Linux distributor or hardware vendor.
- Mount with NFS over TCP and NFS version 3 where possible, and use the largest `rsize` and `wsize` that still provide good performance. Start with the `hard` and `intr` mount options.
- Be sure the network between your clients and filer drops as few packets as possible.
- If you must use NFS over UDP, make sure to follow the instructions documented in "Special network settings" on page 57 for enlarging the transport socket buffers on all your NFS mounts over UDP.

# A

# Enabling utilities and special network settings

## A.1 Special network settings

Enlarging the transport socket buffers your client uses for NFS traffic helps reduce resource contention on the client, reduces performance variance, and improves maximum data and operation throughput.

If a Linux kernel version prior to 2.4.20 is used, the following procedure is necessary because the client does not automatically choose an optimal socket buffer size.

To set proper socket buffer size if using Linux version prior to 2.4.20:

1. Become root on your client.
2. Cd into /proc/sys/net/core.

```
echo 262143 > rmem_max
echo 262143 > wmem_max
echo 262143 > rmem_default
echo 262143 > wmem_default
```

3. Remount your NFS filesystems on the client.

This is especially useful for NFS over UDP and when using Gigabit Ethernet. You should consider adding this to a system startup script that runs before the system mounts NFS filesystems. The size we recommend is the largest safe socket buffer size we have tested. On clients smaller than 16 MB, you should leave the default socket buffer size setting to conserve memory.

Most modern Linux distributions contain a file called /etc/sysctl.conf where you can add changes such as this so they will be executed after every system reboot (Figure A-1). Add these lines to your /etc/sysctl.conf file on your client systems.

*Example A-1: Adding the largest safe socket buffer sizes to /etc/sysctl.conf*

```
net.core.rmem_max = 262143
net.core.wmem_max = 262143
net.core.rmem_default = 262143
net.core.wmem_default = 262143
```

```
[root@itso] cat /etc/sysctl.conf
#Kernel sysctl configuration file for Red Hat Linux
#
#for binary values, 0 is disabled, 1 is enabled.  See sysctl <8> and
#sysctl.conf<5> for more details.

#Controls IP packet forwarding
net.ipv4.ip_forward = 0

#Controles source route verification
netp.ipv4.conf.default.rp_filter = 1

#Do not accept source routing
net.ipv4.conf.default.accept_source_route = 0

#Controls the System Request debugging functionality of the kernel
kernel.sysrq = 0

#Controls whether core dumps will append the PID to the core filename.
#Useful for debugging multi-threaded applications.

kernel.semopm = 100
kernel.core_uses_pid = 1
kernel.shmmax = 2147483648
kernel.shmmni = 4096
Kernel.shmall = 2097152
net.core.rmem_default=262143
net.core.wmem_default=262143
net.core.rmem_max=262143
net.core.wmem_max=262143
fs.file-max = 65536
net.ipv4.ip_local_port_range = 1024 65000
kernel.semopm = 100
#rmem_default=252144
#wmem_default=262144
#rmem_max=262144
#wmem_max=262144
kernel.sem=250 32000 100 128
```

*Figure A-1: An example of sysctl.conf file*

All Linux kernels later than 2.0 support large TCP windows (RFC 1323) by default. No modification is needed to enable large TCP windows. Window scaling is enabled by default.

Some customers have found the following settings to help performance in WAN and high-performance LAN network environments. Use these settings only after thorough testing in your own environment.

- N series storage system:

```
nfs.tcp.recvwindowsize 2097152
nfs.ifc.xmt.high 64
nfs.ifc.xmt.low 8
```

- # Linux NFS client:

```
net.core.rmem_default=65536
net.core.wmem_default=65536
net.core.rmem_max=8388608
net.core.wmem_max=8388608
net.ipv4.tcp_rmem = 4096 87380 8388608
net.ipv4.tcp_wmem = 4096 65536 8388608
```

- The following is in pages, not bytes:

```
net.ipv4.tcp_mem = 4096 4096 4096
```

Usually the following setting is the default for common GbE hardware:

```
ifconfig eth <dev> txqueuelen 1000
net.core.netdev_max_backlog=3000
```

Linux 2.6 kernels support advanced TCP algorithms that may help with WAN performance. Enabling tcp_bic or tcp_westwood can have some beneficial effects for WAN performance and overall fairness of sharing network bandwidth resources among multiple connections (Example A-2).

*Example A-2: Enabling TCP settings*

```
net.ipv4.tcp_bic=1
net.ipv4.tcp_westwood=1
```

Linux 2.4 kernels cache the slow start threshold in a single variable for all connections going to the same remote host. Therefore, packet loss on one RPC transport socket will affect the slow start threshold on all sockets connecting to that server. The cached value remains for 10 minutes.

To flush the cache, you can use (as root):

```
sysctl –w net.ipv4.route.flush=1
```

This might be necessary in a case where some network conditions caused problems, but have been cleared. Cached TCP ssthresh values will prevent good performance for ten minutes on new connections made after the network problems have been cleared up.

**Note:** Slow-start threshold (ssthresh) state variable is used to determine whether the slow-start or congestion avoidance algorithm is used to control data transmission. The slow-start algorithm is used at the beginning of a transfer, or after repairing a loss detected by the retransmission timer, to slowly probe the network to determine its available capacity.

If you experience ARP storms, this could be the result of client or filer ARP caches that are too small. A reasonable workaround is to use routers to reduce the size of your physical networks.

These tuning parameters are documented in the kernel source tree in:

```
Documentation/networking/ip-sysctl.txt
```

## A.2 Controlling file read-ahead in Linux

Read-ahead occurs when Linux predicts that an application may soon require file data it has not already requested. Such prediction is not always accurate, so tuning read-ahead behavior can have some benefit. Certain workloads benefit from more aggressive read-ahead, while other workloads perform better with little or no read-ahead. By default, a Linux 2.4 kernel will attempt to read ahead by at least three pages, and up to 31 pages, when it detects sequential read requests from an application. Some filesystems use their own private read-ahead values, but the NFS client uses the system defaults.

To control the amount of read-ahead performed by Linux, you can tune the system default read-ahead parameters using the **sysctl** command.

To view your system's current default read-ahead parameters, issue the following command:

```
sysctl vm.min-readahead vm.max-readahead
```

or

```
sysctl -A
```

The `min-readahead` parameter sets the least amount of read-ahead the client will attempt, and the `max-readahead` parameter sets the most read-ahead the client may attempt, in pages. Linux determines dynamically how many pages to read ahead based on the sequentiality of your application's read requests. Note that these settings affect all reads on all NFS filesystems on your client system.

You can increase read-ahead if you know your workload is mostly sequential data or you are trying to improve WAN performance. Do this by specifying, as root, the following parameters:

```
sysctl –w vm.max-readahead=255
sysctl –w vm.min-readahead=15
```

This will set your system's read-ahead minimum and maximum to relatively high values, allowing Linux's read-ahead algorithm to read ahead as many as 255 pages. This value takes effect immediately. Usually the best setting for min-readahead is the number of pages in rsize, minus one. For example, if your client typically uses rsize=32768 when mounting NFS servers, you should set min-readahead to 32767. You can add this to the /etc/sysctl.conf file on your client if it supports this; see the section above for details.

**Note:** The 2.6 Linux kernel does not support adjusting read-ahead behavior via a sysctl parameter.

On client systems that support a database workload, try setting the minimum and maximum read-ahead values to one or zero. This optimizes Linux's read-ahead algorithm for a random-access workload and prevents the read-ahead algorithm from polluting the client's data cache with unneeded data. As always, test your workload with these new settings before making changes to your production systems.

## A.3 Enabling trace messages

Sometimes it is useful to enable trace messages in the NFS or RPC client to see what it does when handling (or mishandling) an application workload. Normally you should use this only when asked by an expert for more information about a problem. You can do this by issuing, as root, the following commands:

```
sysctl –w sunrpc.nfs_debug=1
sysctl –w sunrpc.rpc_debug=1
```

Trace messages appear in your system log, usually /var/log/messages. To disable these trace messages, echo a zero into the same files. This can generate an enormous amount of system log traffic, so it can slow down the client and cause timing-sensitive problems to disappear or change in behavior. You should use this when you have a simple, narrow test case that reproduces the symptom you are trying to resolve. To disable debugging, simply echo a zero into the same files with the following commands:

```
echo 0 >/proc/sys/sunrpc/nfs_debug
echo 0 >/proc/sys/sunrpc/rpc_debug
```

To help the syslogger keep up with the log traffic, you can disable synchronous logging by editing /etc/syslog.conf and appending a hyphen in front of /var/log/messages (Figure A-3). Restart the syslog daemon to pick up the updated configuration.

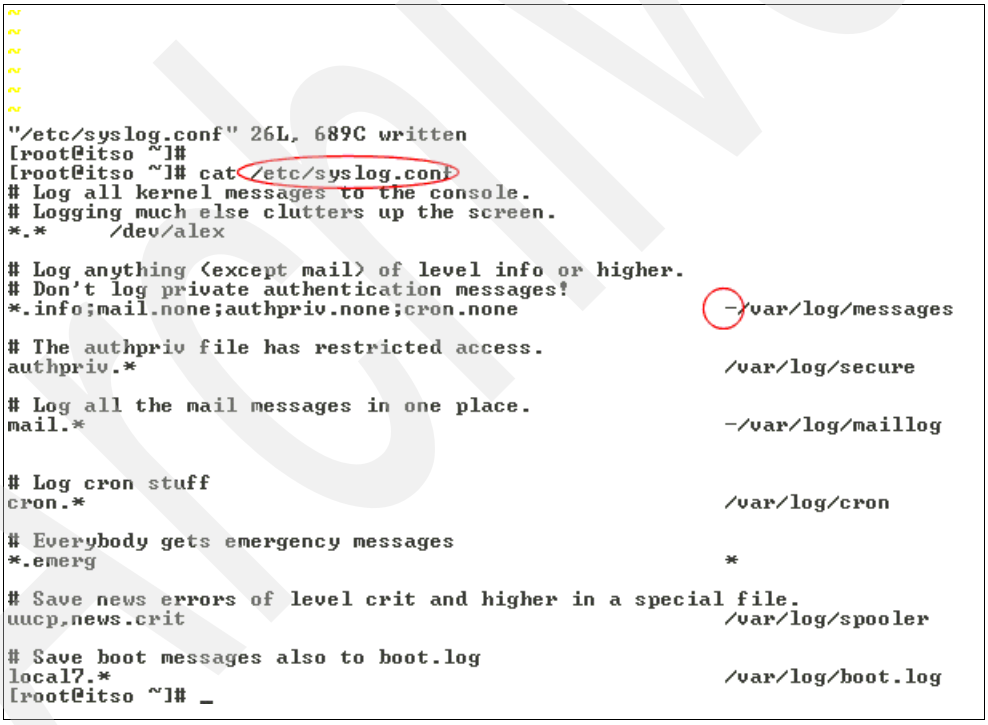


*Figure A-3 Verify synchronous logging disabled after editing /etc/syslog.conf file*

## A.4 Enabling uncached I/O on RHEL AS 2.1

Red Hat Enterprise Linux Advanced Server 2.1 Update 3 introduces a new feature that is designed to assist database workloads by disabling data caching in the operating system. This new feature is called *uncached NFS I/O* and is similar to the NFS O_DIRECT feature found in Enterprise Linux 3.0 and SUSE's SLES 8 Service Pack 3. When this feature is

enabled, an application's read and write system calls are translated directly into NFS read and write operations. The Linux kernel never caches the results of any read or write, so applications always get exactly what is on the server.

Uncached I/O affects an entire mount point at once, unlike NFS O_DIRECT, which affects only a single file at a time. System administrators can combine mount points that are uncached (for example, for shared data files) and mount points that cache data normally (for example, for program executables or home directories) on the same client. Also unlike NFS O_DIRECT, uncached I/O is compatible with normal I/O. There are no alignment restrictions, so any application can use uncached I/O without modification.

When uncached I/O is in effect, it changes the semantics of the "noac" mount option. Normally the "noac" mount option means that *attribute caching* is disabled. When the uncached I/O feature is in effect, "noac" is changed to mean that *data caching* is disabled. When uncached I/O is not in effect, "noac" mount points behave as before. Uncached I/O is turned off by default.

To enable uncached I/O, follow this procedure:

1. Become root.
2. Start your favorite editor on /etc/modules.conf.
3. Add this line anywhere:

   ```
   options nfs nfs_uncached_io=1
   ```

4. Reboot.

Uncached I/O will take effect after you reboot your client. Only mount points that use the "noac" mount option will be affected by this change.

# Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

## IBM Redbooks

For information about ordering these publications, see “How to get IBM Redbooks documents” on page 64. Note that some of the documents referenced here may be available in softcopy only.

- *Multiprotocol Data Access with IBM System Storage N series,* REDP-4176
- *N Series Snapshot: a Technical Discussion*, REDP-4132
- *IBM System Storage N series File System Design for an NFS file server*, REDP-4086
- *IBM System Storage N Series Implementation of RAID Double Parity for Data Protection*, REDP-4169

## Other publications

These publications are also relevant as further information sources:

- *IBM System Storage N series Data ONTAP 7.2 Storage Management Guide,* GC26-7966-00
- *IBM System Storage N series Data ONTAP 7.2 File Access and Protocols Management Guide*
- *IBM System Storage N series Data ONTAP 7.2 Software Setup Guide,* GC26-7975-01

## Online resources

These Web sites are also relevant as further information sources:

- Support for Data ONTAP

  `http://www-304.ibm.com/jct01004c/systems/support/supportsite.wss/supportresources?taskind=3&brandind=5000029&familyind=5329797`

- Support for IBM System Storage and TotalStorage products

  `http://www-304.ibm.com/jct01004c/systems/support/supportsite.wss/storageselectproduct?brandind=5000029&familyind=0&oldfamily=0&continue.x=3&continue.y=8`

- Network attached storage (NAS)

  `http://www-03.ibm.com/systems/storage/nas/index.html`

- Tcpdump home page

  `http://www.tcpdump.org`

- Ethereal home page

http://www.ethereal.com

- Linux NFS FAQ and how-to

  http://nfs.sourceforge.net

- Linux network boot loader information

  http://syslinux.zytor.com

- Linux automounter information

  http://www.spack.org/index.cgi/AutomountHelp?action=show&redirect=LinuxAutomounter

- Linux NFSv4 and NFS with Kerberos development

  http://www.citi.umich.edu/projects/nfsv4/

- Network Time Protocol daemon home pages

  http://www.ntp.org

## How to get IBM Redbooks documents

You can search for, view, or download books, papers, Technotes, draft publications and Additional materials, as well as order hardcopy books, at this Web site:

**ibm.com**/redbooks

## Help from IBM

IBM Support and downloads

**ibm.com**/support

IBM Global Services

**ibm.com**/services

# Index

## L



## M



## N



## O



## P



## R



## S

## T



## U



## V



## W

(0.1”spine)
0.1”<->0.169”
53<->89 pages

# Using the Linux NFS Client with IBM System Storage N series

Redbooks

**Kernel releases and mount options**

**Tuning the Linux client for performance and reliability**

**Utilities to support advanced NFS features**

This book will help you get the best from your Linux NFS clients when used in an environment that includes IBM System Storage N series products. It begins with a general discussion of Linux distributions and kernels, and includes a comparison of some of the features of the 2.4 and 2.6 kernels. A detailed discussion of mount options and network protocols is presented to guide you through configuration tasks. Procedures for tuning your Linux clients and steps to diagnose performance and reliability problems are presented, along with information about additional resources you can consult if necessary. Finally, steps for configuring and using utilities that support advanced NSF features are provided.

This document is appropriate for customers, systems engineers, technical marketing engineers, and customer support personnel who install and configure Linux systems that use NFS to access N series systems and network caches.

**INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION**

**BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE**

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

**For more information:**
**ibm.com**/redbooks

SG24-7462-00 ISBN 073848928X