Proceedings of the Linux Symposium

Volume Two

July 19th–22nd, 2006
Ottawa, Ontario
Canada

NFSv4 Test Project

Aurelien Charbon
Tony Reix
Vincent Roqueta
Bull SAS
tony.reix@bull.net

Bryce Harrington
OSDL
bryce@osdl.org

J. Bruce Fields
CITI
bfields@fieldses.org

Trond Myklebust
Network Appliance, Inc.
Trond.Myklebust@netapp.com

Suresh Jayaraman
Novell
sjayaraman@novell.com

Jeff Needle, Barry Marson
Red Hat
jneedle@redhat.com, bmarson@redhat.com

Abstract

This paper presents the testing effort done around NFSv4[1] by the Linux NFSv4 community. As an introduction, we explain the rationale for such a heavy testing activity, why NFSv4 was needed, the current status of NFSv4, and some Use Cases. Chapter 3 describes the tools used for testing integral features of NFSv4 in the areas of functionality, interoperability, robustness, performance, and security: where they come from, and which parts of NFSv4 they are aimed to test. We also describe some tools used for analyzing problems and loads. Chapter 4 first explains the goals of the NFSv4 testing team and how contributors are working together. Major events for NFSv4 since January 2004 are displayed in a developmental timeline. Then, four contributors (OSDL, Bull, Novell, Red Hat) amongst many others describe in details their NFSv4 testing activity, explaining what they have already done and what their future plans are. OSDL and Bull are contributing to the continuous testing activity of fresh kernel+CITI versions, though Novell and Red Hat test NFSv4 in the eco-system of their distributions. As a conclusion, the paper shows that the testing efforts have generated significant improvements in all the test areas and that the core of Linux NFSv4 is stable and powerful. Also, some ideas are presented about the future of NFSv4 protocol and of Linux NFSv4.

[1]Network File System version 4.

1 Introduction

NFS Version 4 adds a number of powerful new features to address NFS shortcomings in security, migration, performance and other areas. In order to make NFSv4 the new industry standard for Linux, these features must be thoroughly and frequently tested to ensure they are functional, robust, efficient, and secure. The goals for testing NFSv4 on Linux are to make it more stable, more mature, more interoperable with other NFS implementations, and to improve the entire ecosystem of software that interacts with NFS.

NFSv4 for Linux has been under development at CITI and NetApp since 2001. This Linux NFSv4 testing task force started in 2004 with participants from OSDL, Bull, IBM, NetApp, Novell, and Red Hat, plus many other companies and individual contributors.

Because NFSv4 is a complex and critical infrastructure service, the testing is both very important and very challenging. Our key strategy in achieving our goals has been a large and collaborative task force that focuses on different testing approaches, sharing results and working directly with the developers to validate fixes.

Development of Linux NFSv4 follows the Open Source *Release Early, Release Often* model. This means that new features for the Linux implementation of the NFSv4 protocol become available in the mainline kernel as soon as they're ready. Ultimately, the new features will enable or enhance a number of Use-Cases including high performance computing clusters, large scale render farms, massive corporate provisioning, and secure Intranet and Internet file sharing.

OSDL's role is to facilitate this testing through establishing a testing community around Linux NFSv4. Initially, this involved planning activities such as collaborating with stakeholders in creation of a *Test Matrix/Wiki* itemizing and prioritizing testing needs, and in providing opportunities for members of the community to meet and collaborate. Recently, OSDL's activities focus on participating in designing and running tests for regression and installation testing, and for checking configuration robustness.

Terminology

Before getting into details about testing it is useful to clearly define the terminology we have adopted in testing NFSv4.

First, one may want to check that NFSv4 features work as they have been designed for. Usually each function is tested separately to cover the whole function set. It is: *functional testing*.

Second is *interoperability testing*, which involves comparing the Linux NFSv4 implementation against other implementations, as well as reviewing how Linux NFSv4 interacts with other components in the system.

Then, one may want to check that NFSv4 still continues to work under high load, often with random and simultaneous operations. This aims at generating extreme cases, not reachable with functional tests, to stress the system. It is: *robustness testing*.

Since many people need to know how many clients can be connected to a NFSv4 server, one must measure how long NFSv4 can deliver some service or how many actions can be managed in parallel and in a defined period of time: *performance testing*. Performance testing consists in analyzing figures (speed, time, CPU load, Memory load, etc.) and in trying to determine where the bottlenecks are.

Security testing may have different meanings. In the current case the goal of testing NFSv4 security is not to test that security tools used by NFSv4 (like Kerberos) work well. Instead,

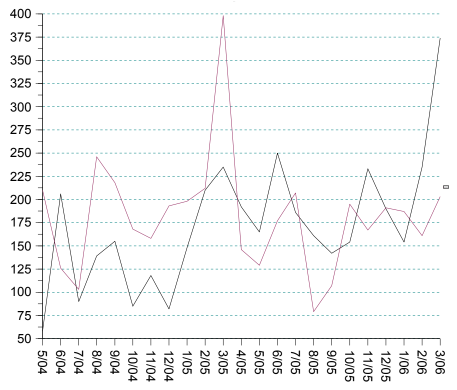

Figure 1: `nfsv4` and `nfs` mailing list activity.

NFSv4 security testing first checks that NFSv4 still works fine under a security environment like Kerberos, and then it checks that NFSv4 server does not reduce the security of Linux by opening security holes malicious or dangerous people could use.

`nfsv4` and `nfs` mailing lists

Problems dealing with Linux CITI NFSv4 are discussed on the `nfsv4@linux-nfs.org` mailing list, though general Linux NFS discussions about development and interoperability are discussed on the `nfs@lists.sourceforge.net` mailing list. Figure 1 shows the activity of these 2 mailing lists since may 2004: `nfsv4` mailing list (in black), which has more than two hundred different participants, now has about the same activity than the `nfs` mailing list (in red). As expected, the most talkative people on `nfs4` mailing list are: Bruce Fields, Bryce Harrington, Trond Myklebust, Kevin Coffman, and then Vincent Roqueta.

2 NFSv4 Description

2.1 Why NFSv4 ? What is new in NFSv4 ?

NFSv4 brings a number of improvements to NFS.

NFSv2 and NFSv3 do not themselves include file locking or ACL[2] management; instead, those functions are performed by separate RPC[3] protocols. In addition, a mount protocol is required to obtain the root file-handle of an exported filesystem. Thus four distinct RPC protocols are required for full functionality.

NFSv4 integrates all of these into the same protocol, simplifying firewall management and enabling previously unsupportable features, such as mandatory file locking.

For security, NFS implementations have traditionally relied on private networks and locked-down clients. The `rpcsec_gss` protocol adds support for cryptographic security using per-user credentials, thus eliminating the need to trust every host on the network. While NFSv2 and NFSv3 can also take advantage of `rpcsec_gss`, NFSv4 is the first to require implementation (not use) of `rpcsec_gss`, and NFSv4 integrates it into the protocol more thoroughly.

NFSv4 also allows servers to hand out *delegations* to clients, giving the client shared (read-only) or exclusive (read and write) access to the file, for improved caching.

NFSv4 provides some support for filesystem replication and migration with a new *fs_locations* attribute that clients can use to find other servers exporting the same filesystem data.

[2]Access Control List.

[3]Remote Procedure Call.

NFSv4 enables better support for Windows APIs with open share locks and a fine-grained ACL model based on Windows.

The last two NFSv4 features are more subtle, but together add an important layer of extensibility.

First, NFSv4 brings in a mechanism for introducing incremental updates to the protocol, called *minor versions*. Draft specifications and prototypes for minor version 4.1 are currently available; see chapter 5 (Future of NFSv4) on page 292 for details.

Second, NFSv4 operations are now sent as series of smaller operations, called *compound RPCs*. For example, what an NFSv3 client would do with a single NFSv3 WRITE operation might be accomplished by an NFSv4 client using a compound RPC consisting of the three operations: PUTFH (to indicate which file the following operations apply to), WRITE, and GETATTR (to update the client's attribute cache). The compound RPC adds flexibility to the protocol, especially when extending it, since new operations can combine with existing operations in interesting ways.

Finally, while NFS has always been a freely documented and widely implemented protocol, previous protocol specifications have been the work of Sun Microsystems. NFSv4 is the first version that is actually developed within the IETF[4] and hence whose development is also open. In practice we have seen wide and fruitful participation in the process.

2.2 Status of NFSv4 on Linux

The Linux NFSv4 client and server implementation supports all of the basic features of NFSv4. In more detail:

[4]Internet Engineering Task Force.

Delegations are supported, and the client will take advantage of a file delegation where possible to perform `opens` and `closes` without contacting the server. Work is underway to provide even more aggressive caching on the client, if desired, using on-disk data caching. The server implements delegations using leases on the exported filesystem, allowing it to cooperate correctly with local and Samba users.

File locking is supported, and locks can be handled entirely on the client when delegations allow.

ACLs are supported, though client-side ACL-manipulation tools are still under development, and server-side support is limited by the need to store NFSv4 ACLs as less fine-grained POSIX[5] ACLs.

Kerberos-based security is fully implemented, and support for the public-key based SPKM3[6] and LIPKEY[7] mechanisms is under development.

We have patches implementing preliminary support for replication and migration; further work needs to be done to refine them and integrate them into mainline.

Ongoing development includes stabilization and tuning. We are also interested in improving NFS (especially NFSv4) support for cluster filesystem exports; for example, we need to ensure that locks acquired by an NFS server on one node of a cluster can be enforced by an NFS server on another node.

Server user interfaces are under revision to provide a more consistent interface for users upgrading from NFSv2 and NFSv3.

[5]Portable Operating System Interface

[6]Simple Public-Key Mechanism.

[7]Low Infrastructure Public Key.

2.3 Use Cases

The new features in the NFSv4 protocol are intended to improve performance and reliability for proved usage scenarios. While we cannot enumerate every possible use, for testing purposes we've identified several distinct use cases where NFSv4 would be expected to show benefit over previous versions.

Scientific computing cluster. Laboratories use NFS to communicate between the nodes in a large computational cluster. This usage scenario involves days or weeks of quiescence, with occasional bursts of heavy read activity, followed by intensive write operations. This use case will benefit from robust network recovery and good write performance.

Render farms. This use case is in a way the inverse of the scientific cluster. Render farms also involve a large number of computational nodes, but the write operations are more continuous over time, punctuated by intensive read operations. NFSv4's delegation and caching improvements may be the biggest benefits in this use case.

Provisioning. A number of large users use network filesystems for deployment of software or software updates, either to large server installations or to large workstation deployments. In either case, issues include congestion control (such as if many machines attempt to access server resources simultaneously), access control, and migration and replication.

Databases. Use of Network Attached Storage (NAS) and similar technologies often results in designs that place a database back-end on an NFS share. This usage provides benefits including backup/rollback and flexible volume management, but can raise performance concerns. New features of NFSv4 worth exploring with this use case are delegations and caching improvements.

3 Testing Tools

Stability and robustness:

First and foremost, NFSv4 acts as a filesystem. So, the main priority in NFSv4 testing is checking that the filesystem is stable and robust. Many generic, Open Source filesystem test tools are available to perform such tests; many of them, including several listed below, are part of the LTP[8] [10].

IOZone [9] was originally a performance tool. Developed by IOZONE.ORG, ORACLE and HEWLETT PACKARD, it measures raw throughput of file operations such as *read*, *write*, *reread* or *rewrite*. These throughputs are measured with various file sizes and *read/write* sizes. A typical IOZone test has a total of 1430 measurements. IOZone allows mounting and unmounting filesystems with various parameters between tests. It can be used to stress mount program with various parameters.

Fsx [10] is an APPLE COMPUTERS in-file stress program available in LTP. It performs the following file operations : *mapped read, mapped write, read, write, truncate*. FSX checks if data corruption occurred during these operations.

Fsstress is FSX's complement, and is also available in LTP. It was written by SILICON GRAPHICS INC.. It stresses filesystem tree structure by doing random operations on the tree structure: file creation, recursive directories, symlinks manipulations.

Locks tests [10] launch multiple processes on multiple clients. This tool was designed

[8]Linux Test Project.

by BULL SAS to stress NFSv4 locks, and contributed to LTP. Processes try to perform locks-related operations on the same file or file section. Results are compared to the expected results. It can be used to stress both network and local filesystems.

ACL tests [10] were written by BULL SAS in order to stress ACLs within NFSv4, and are available in LTP. The test suite creates numerous users and ACL rules and combines them; then it checks that actual accesses match ACL rules.

Connectathon 2004 [8] was designed by SUN MICROSYSTEMS, INC to perform interoperability testing of critical operations. It performs high-level operations that often reveals interoperability troubles.

FFSB [12] is a versatile and useful filesystem test. It was created and enhanced for NFSv4 by IBM. It can both stress the whole filesystem, mimic various load profiles and collect test information.

NetEm [13] is a kernel component developed by Steve Hemminger at OSDL (available when configuring the kernel) that allows to modify network behavior. It provides the following features:

- dynamic delay between packets (RTT).
- packet loss, duplication, corruption, re-ordering, collisions.
- rate control, and non-FIFO queuing.

It can be easily configured to mimic the behavior of real networks.

NewPyNFS [14] is a Python-based test-suite developed and maintained by the CENTER FOR INFORMATION TECHNOLOGY INTEGRATION (CITI) of the University of Michigan. Unlike the tests described above, it is not a *black box* test tool: it implements a python NFSv4 client and server. It was specifically designed to test NFSv4 features, including ones that are not yet fully implemented.

Analysis tools

OProfile is a kernel module used to collect statistics of CPU load by function.

Ethereal [11] is a well known network analyzer. It helps checking that NFSv4 server and clients exchange valid sequences of information over the network.

Needed tools

Since new NFSv4 features should appear soon within Linux NFSv4, appropriate new tests will be needed.

Migration and Replication is a relatively new feature, and a functional/robustness test will be necessary. This test would emulate or demonstrate the transfer of an NFSv4 share from one NFS server to another, and it would check that the client is able to continue accessing the data seamlessly, while measuring the impact on the client during the transition.

New tests will be needed to exercise full NFSv4 ACLs, Named Attributes, and Directory Delegation.

We also need tests of the security of NFSv4. We need to measure the impact on performance of running NFSv4 with security and evaluate the robustness of NFSv4 when attacked.

4 NFSv4 Tests

Hereafter we present the testing effort done by four contributors: OSDL, Bull, Novell and Red Hat. Many other companies and people have contributed to improve NFSv4, testing it, providing patches, warning about mistakes, or providing information about oops and bugs they are experiencing in their labs.

NFSv4 is being tested in three different ways: 1) OSDL and Bull have developed tests and are using them plus others (filesystem tests, LTP) to continuously check that no regression occurs; 2) many contributors or early adopters are using NFSv4 in their own complex and specific environment; 3) Novell and Red Hat are hardening NFSv4 in the environment of their future distributions by using regression tests. These different approaches are complementary. Regression and stress tests enable to verify that NFSv4 core is reliable in a clean and standard environment. And tests done in various and unique environments enable to check that the whole NFSv4 ecosystem is robust and to clean NFSv4 of all little mistakes in its childhood.

The timeline on page 282 presents the evolution of NFSv4 since beginning of 2004. Vertical bars on the right show how main features of NFSv4 have evolved, from bad (red) to good (green). Main events appear in the middle. Notice that regressions appear from time to time and are quickly fixed. Also notice how long it took to make locks reliable.

4.1 OSDL

The Open Source Development Labs (OSDL) [15] became involved in the NFSv4 testing effort at the request of the NFSv4 community and through OSDL's Data Center Linux (DCL) initiative. Initial involvement included assisting in organizing efforts, identifying test plans, establishing testing priorities, and facilitating discussion between companies and community members. Today OSDL's role is in conducting regression testing of all kernel patch releases by the NFSv4 community, and ancillary activities to help facilitate and promote use of NFSv4.

The test matrix [16] is a listing of test tasks that were felt to be needed to fully test the Linux NFSv4 system, broken down into the following categories: Functional, Robustness, Performance, Interoperability, and Security. Through discussions with testers and developers, these tasks were prioritized, and an NFSv4 Testing Road-map was generated, itemizing the high priority testing tasks and identifying which organizations will be performing which tasks. OSDL signed up for several Functional/Robustness testing tasks involving regression testing and cross-compile testing.

Cross-compile testing is done on every kernel patch released by CITI using the *Patch Lifecycle Manager* (PLM). These builds target a number of different architectures, including i386, x86_64, ppc, ppc64, sparc, sparc64, arm, and alpha. Compiles are performed against several different configs, including allyesconfig, allmodconfig, allnoconfig, and defconfig. *Sparse* has also recently been added. These builds have been useful in identifying both issues particular to certain platforms (such as only the ppc architecture, or only 64-bit systems), as well as variation in config settings. For example, the developer may only be checking his work with a given config setting turned on, and may have accidentally added an issue that only shows up with that config setting turned off; this cross compile system will have a better chance of catching these classes of defects.

Regression testing is done using *Crucible*, a framework to automatically download new patches from CITI and kernels from

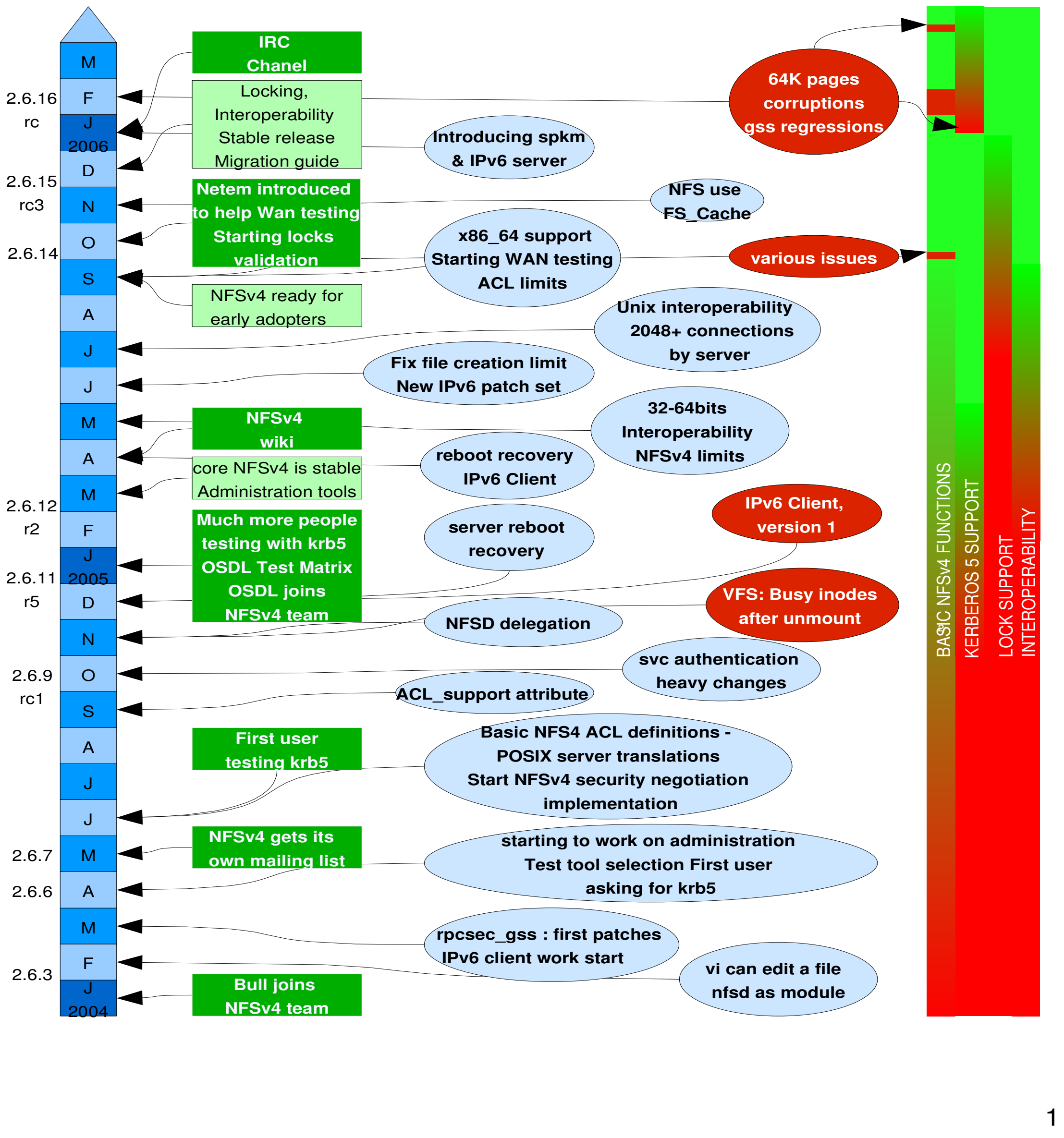
M
F
J 2006
D
N
O
S
A
J
J
M
A
M
F
J 2005
D
N
O
S
A
J
J
M
A
M
F
J 2004
2.6.16 rc
2.6.15 rc3
2.6.14
2.6.12 r2
2.6.11 r5
2.6.9 rc1
2.6.7
2.6.6
2.6.3
IRC Chanel
Locking, Interoperability Stable release Migration guide
Netem introduced to help Wan testing Starting locks validation
NFSv4 ready for early adopters
NFSv4 wiki
core NFSv4 is stable Administration tools
Much more people testing with krb5 OSDL Test Matrix OSDL joins NFSv4 team
First user testing krb5
NFSv4 gets its own mailing list
Bull joins NFSv4 team
64K pages corruptions gss regressions
Introducing spkm & IPv6 server
NFS use FS_Cache
x86_64 support Starting WAN testing ACL limits
various issues
Unix interoperability 2048+ connections by server
Fix file creation limit New IPv6 patch set
32-64bits Interoperability NFSv4 limits
reboot recovery IPv6 Client
IPv6 Client, version 1
server reboot recovery
VFS: Busy inodes after unmount
NFSD delegation
svc authentication heavy changes
ACL_support attribute
Basic NFS4 ACL definitions - POSIX server translations Start NFSv4 security negotiation implementation
starting to work on administration Test tool selection First user asking for krb5
rpcsec_gss : first patches IPv6 client work start
vi can edit a file nfsd as module
BASIC NFSv4 FUNCTIONS
KERBEROS 5 SUPPORT
LOCK SUPPORT
INTEROPERABILITY
1

Figure 2: Linux/NFSv4 History.

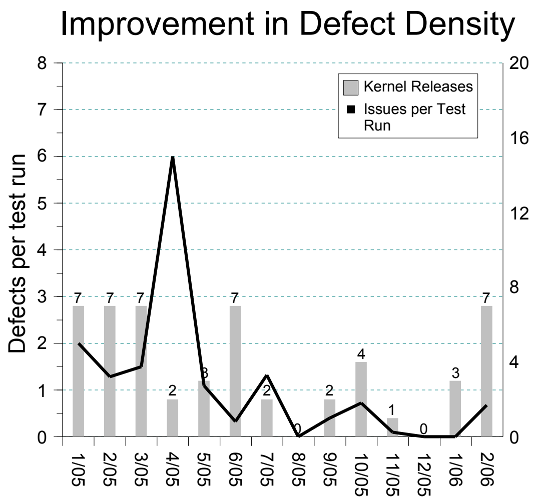

Figure 3: Defects.

`kernel.org`. It then patches and compiles the kernel on a client and a server, boots them to that kernel, and then runs a sequence of tests (cthon04, NewPyNFS, IOZone, and LTP) on them. The results are collected, parsed, and analyzed for abnormalities or other unusual behaviors. These are reported to the developers, and efforts are taken to identify the root causes where they are not obvious.

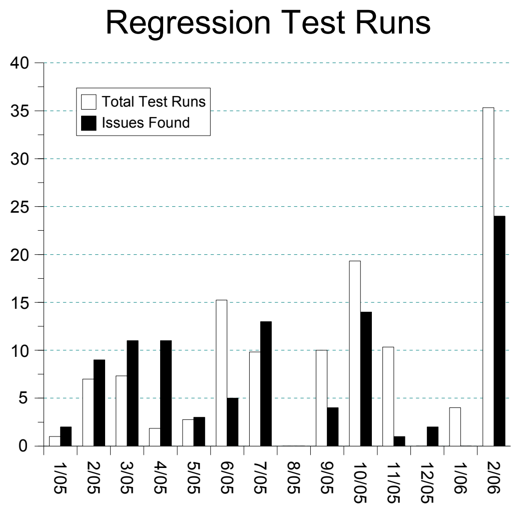

Figure 4: Regression Tests.

Typically, most issues the regression testing finds is via the NewPyNFS test, such as changes in return codes from functions affected by recent development changes. In some cases these identify legitimate defects, but in other cases they simply indicate areas where different people have interpreted the spec in different ways; even this is useful because it identifies areas where further discussions about the spec are needed, to resolve what should happen. In this latter case it is not uncommon for the test suite to require modification to reflect the new consensus.

The biggest challenges in establishing the automated framework is boot control. Invariably the client or the server will hang. It is necessary to use a watchdog process to automatically detect that the machine has become inactive (such as failure to respond to pings at a time when it should respond), and then perform a remote power cycle on it. As well, there is always a chance that a given kernel will not boot; to account for this situation, the boot-loader's default kernel is kept to a static, known-good kernel, and the kernel-to-test is specified via `lilo -R`.

The hardware included in the OSDL test framework is primarily x86 based systems running Gentoo Linux, but also includes a NetApp filer, an amd64, a ppc64, and an itanium 2. Other hardware may be added in the future, depending on donations to the lab. The principle challenge to integrating new hardware is automating boot control; the system must support both some form of remote power management (in worst case, through use of a separate interruptible power unit), and a boot-loader mechanism to test a new kernel and fallback to a known good one on next reboot. Serial console access and/or logging is also important for catching console errors.

OSDL future work

Due to the success seen through use of the regression test framework, it is expected that this will be expanded with more tests, more hardware, and more ways to put tests into complex configurations. For example, to date the automated boot mechanisms have only been used between tests to *reset the system*, however it could be invaluable to boot the server or client during a test, and double-check how the system as a whole responds. Indeed, there is test code in both LTP and NewPyNFS intended to check performance through reboots; these test cases are not typically run, for obvious reasons, so this framework could enable us to increase our testing coverage to these areas.

Building on this, network stability testing can be done by introducing perturbations in the network such as network partitions, dropped or corrupted packets, and so forth.

To help promote the advantages of adopting NFSv4, it would be worthwhile to add some performance comparison capabilities to the test harness. One idea is to simply perform timed kernel builds over NFSv3 and NFSv4 mounts, and compare performance. Another idea is to create a 3D graphics render-farm using a server and multiple clients using POV-Ray and the POV-Any software, and to time the performance of distributed renders. Ideally, these should illustrate how NFSv4's delegations and other performance features impact real world workloads.

4.2 Bull

Bull's contribution to the Linux NFSv4 project started in January of 2004.

When problems are found, either we directly expose them to the NFSv4 developers or we talk on the `nfs4` mailing list or we open a new bugzilla ticket, depending on the complexity of each problem.

Each time a task is finalized (like: regression tests on last kernel-CITI version, performance measures, ...) we publish a *News* on our website [21].

We are using a limited number of machines: four ia32 machines with two processors used both for Linux testing and for Solaris interoperability, and two PowerPC machines used both for Linux testing and for AIX interoperability. One ia32 machine is a x86_64 machine and is used for 64bits testing, complementing testing with AIX 64bits. All machines are connected with GigaBits boards and switches, for performance measurement purposes. Machines are installed with different Linux distributions (FedoraCore from Red Hat, SLES from Novell, and Debian).

4.2.1 Regression tests

Each release of the kernel and of the CITI patchset is exercised with tests in order to detect regressions in stability, robustness and performance areas. In most cases, bugs are the result of new patches. The goal is to ensure that these bugs will be corrected in the next CITI_NFS_ALL patch.

The following tests are run, using different tools:

- Connectathon 04 testing suite: do basic conformance testing.
- Two hours FSSTRESS and FSX tests: check robustness.
- IOZone: compare performance with previous versions.
- LTP Locks and ACL tests.

Once a problem is discovered, a new bug report is created in the Linux NFSv4 Bugzilla [18]. Then, Linux NFSv4 developers at CITI are warned directly or via the `nfs4` mailing list. They are provided with a stack trace if a kernel Oops occurred or a network trace in other cases.

Tests are run with two different underlying local filesystems: ext3 and ReiserFS.

Regression tests are also used to detect interoperability issues. The tests are performed on different Linux platforms (ia32, x86_64 and ppc) and with non-Linux client or server.

4.2.2 Stress and robustness

As a **Network File System**, NFSv4 provides two types of functions and mechanisms:

Filesystem functions. Since NFSv4 mimics the behavior of a local filesystem to applications, it provides common filesystem related functions, such as *read, write, open, mkdir* ... but also more advanced functions such as *fcntl, flock, acl* ...

NFSv4 specific functions. NFSv4 tries to appear as a local filesystem, but it is not. It provides several functions and mechanisms over the network, and more specifically over Internet. These functions include: *gss support, delegation, automounter, security negotiation, migration, replication* ...

In order to ensure the stability and robustness of NFSv4, all its functions have to be stressed.

Core filesystem functions

The first step when testing NFSv4 deals with checking its stability on the main architecture: ia32. Several tests are performed on core NFSv4 functions. These functions are common to all filesystems. They provide basic interfaces for in-file manipulation functions (*open, read, write,* ...) and filesystem tree manipulation functions (*mkdir, ls, ln*).

The first goal of testing is to ensure that no corruption of data occurs when manipulating files. The FSX stress tool is quickly run, and it must be successful. It is used to prove that Linux NFSv4 does not corrupt data.

The second goal of testing is to ensure that file manipulations are correctly handled, especially when using very long path or very deep sub-tree structures and sub-directories with multiple processes. When we started to use FSSTRESS it helped to reveal numerous problems in many areas (symlink overflow, deadlock or memory leak) that are reliable now in the current versions of Linux NFSv4.

One year after we started using FSSTRESS (in April 2005) Linux NFSv4 was able to sustain the concurrent load of 10 processes during 24 hours, without any problem. Three months later, NFSv4 reached 72 hours of stress under FSSTRESS, without any bugs. From this date, NFSv4 filesystem tree manipulation is considered to be stable.

Locks

When we started to stress lock features in early 2005, there was only one tool available to test locks: Connectathon. We have used Connectathon during some months to stabilize NFSv4 locks implementation. It helped to find many bugs on several architectures.

However, Connectathon was not designed to perform heavy stress operations. So, a new test tool was required, that we designed and imaginatively named: *LockTester*. This lock test launches an arbitrary number of processes on one or more clients to heavily stress the NFSv4 server and client. Processes try to perform various lock-related functions on the file at the same time. This tool has helped to find several complex bugs, and it is recommended to use it when running regression tests. It has been successfully integrated into the `LTP` suite, in `network/nfsv4/locks` branch (however, it can be used with any filesystem).

By the end of 2005, one NFSv4 server was able to manage more than 500 local concurrent processes, and more than 2000 concurrent processes launched from four machines (x86, ppc).

High Load

In order to over-stress NFSv4, we have used up to 2048 IOZone processes running on two machines and concurrently loading NFSv4 on a recent kernel: no problem was revealed.

4.2.3 ACLs

NFSv4 defines a flavor of Access Control Lists (ACLs) resembling Windows NT ACLs, described in an IETF Internet-Draft [5]. A number of operating systems use a different flavor of ACL based on a POSIX draft. NFSv4 clients and servers on such operating systems may wish to map between NFSv4 ACLs and their native ACLs. Interoperability tests aim at verifying this mapping.

An ACL test suite built with python scripts and C programs has been added to the Linux Test Project tree. It is now available in the `network/nfsv4` branch.

It aims at to test the following points:

- ACL conformance: verify that actual access conforms to the access control list of the file. It includes conformance testing of ACLs on files and directories, but also on default directory ACLs.
- ACL robustness: multiple clients stress one server with random ACL requests on one single file, or on multiple files.
- ACL limits: determine the maximum length of an ACL. ACL limits tests have been run with Linux, Solaris 10 and AIX 5.3 on server and client sides: no interoperability issues have been found.

The tests are delivered with tools that help managing the thousands users needed by the tests.

Tests have been run with different underlying filesystems: ext3, xfs and ReiserFS.

Main problems found:

The tests have shown that the main current limitation is that the server does not allow the client to retrieve an ACL greater than one memory page, due to the underlying RPC. So, when the name of users and groups appearing in the ACLs are 6 characters long, the limit size is about 35 ACL entries.

ACL Interoperability tests:

There are now three ACL models to deal with: NFSv4, Windows, and "POSIX ACLs"/mode bits. And one must decide what to do with them all in the face of existing users, tools, and system interfaces that assume one or the other. For example: since individual clients and applications with different ACL models may not deal well with the full generality of NFSv4 ACLs, problems may also arise from clients reading and modifying ACLs written by clients with different expectations.

So it is useful to run these ACL tests as long as the ACLs' implementation in NFSv4 code is under development.

Remaining tests:

Interoperability tests with Windows filesystems need to be performed. Also, we have to develop new tests that enable the testing of all the features of NFSv4 ACLs, regardless of the underlying filesystem.

4.2.4 Performance

Most NFSv4 performance measures have been done with IOZone, which is designed to measure a global throughput on a filesystem.

These IOZone tests, combined with vmstat, have been useful to detect performance problems as well as functional ones, such as wrong use of Kerberos 5.

NFSv4 compared to NFSv3 & Samba

NFSv4 (TCP) has been compared to other well known network filesystems: NFSv3 (UDP) and Samba.

Read performance

Figure 5 shows `Read` performance of NFSv3, NFSv4 and Samba 3.

For large files (greater than 4 MegaBytes) and both in asynchronous and synchronous (no cache is used) modes, NFSv4 (red and purple lines) and NFSv3 (green and sky blue lines) have similar performance.

For small files (smaller than 4 MegaBytes) and both in asynchronous and synchronous modes, NFSv4 outperforms NFSv3.

For small files, NFSv4 performance is between 2 and 15 times better than Samba (cobalt blue line). For large files, NFSv4 is over 15 times better than Samba.

Write performance

Figure 6 shows `Write` performance of NFSv3, NFSv4 and Samba 3.

In asynchronous mode with small files, NFSv4 (red line) is about 6 times faster than NFSv3 (green line) and 3 times faster than Samba (cobalt blue line). The cache used by NFSv4 is very efficient for small files.

For large files, NFSv4 (purple line) and NFSv3 (sky blue line) have similar performance.

In synchronous mode with small files, the performance of NFSv4 and NFSv3 is between one third and one half of the performance of Samba. For large files, NFS is about 20% faster than Samba. NFSv4 and NFSv3 show the same performance.

Conclusions

While NFSv4 is still being developed, its performance is similar to NFSv3 performance and it outperforms Samba. For small files, NFSv4 performance clearly outperforms NFSv3.

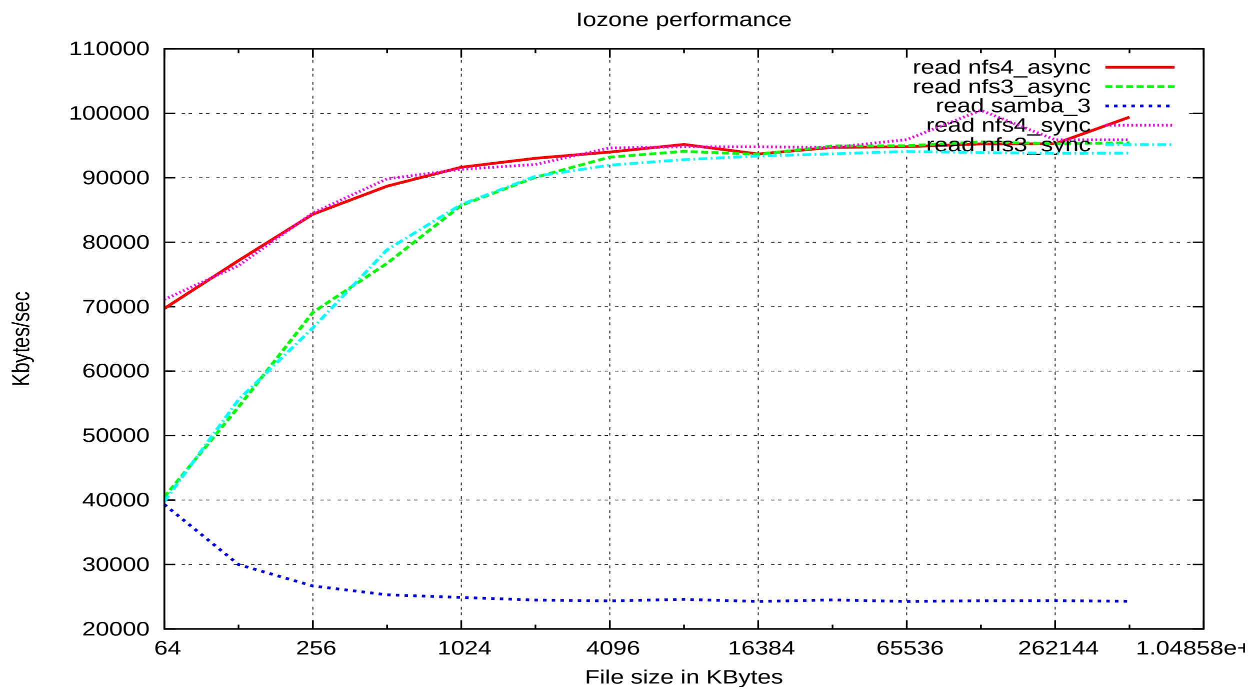

Figure 5: **READ**: NFSv4 vs NFSv3 (synchronous and asynchronous modes) & Samba3.

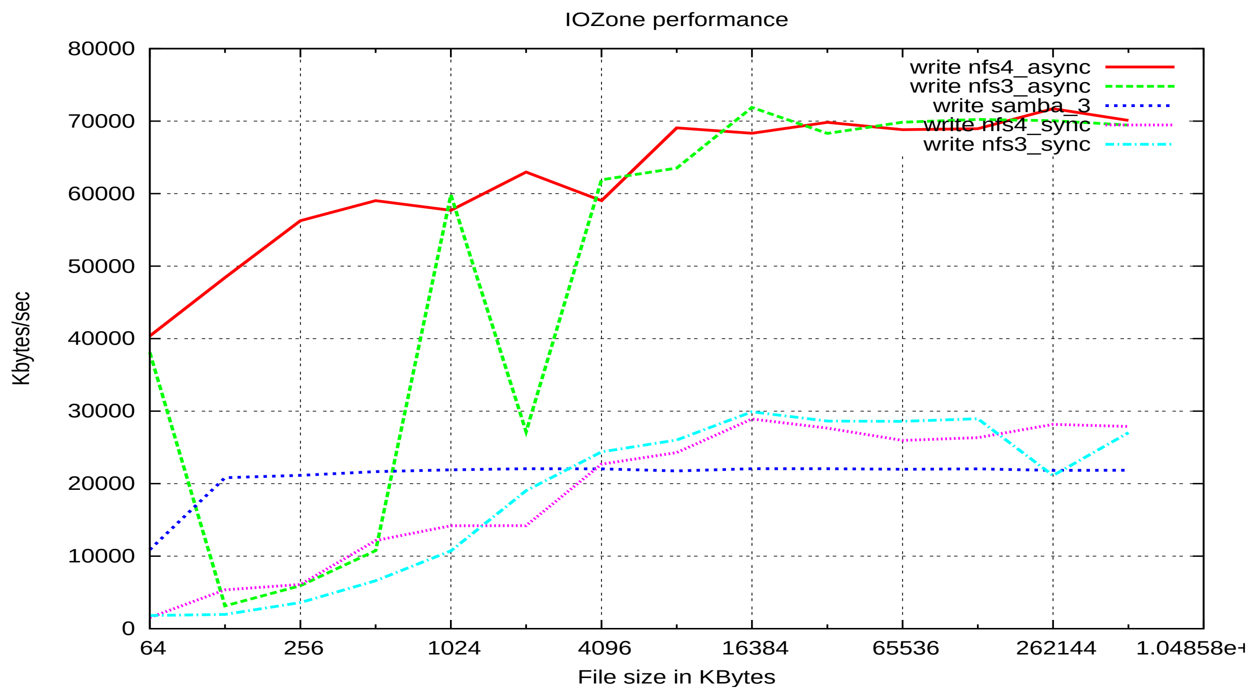

Figure 6: **WRITE**: NFSv4 vs NFSv3 (synchronous and asynchronous modes) & Samba3.

Configuration of Tests

- Network: GigaBit Ethernet
- Machines (client and server): dual ia32 machines
- Kernel: 2.6.15
- Test performed: IOZone standard tests (`-ace -r 32 -U`)

4.2.5 Interoperability testing

Hardware interoperability

Since Linux NFSv4 will be used on different architectures, it is worth checking early if all features of Linux NFSv4 run perfectly on them. We focus on issues generated by 32/64 bits alignment and little/big endian problems. Tests are done on Intel x86, Intel/AMD x86_64, and IBM ppc64 architectures, used either as NSFv4 server or NFSv4 client. A couple of bugs related to these two kinds of problems have already been found and fixed.

These tests deal only with basic features of Linux NFSv4.

Table 1 shows the status of the interoperability tests when run with 2.6.12 kernel. Several problems appeared when running Connectathon 04, showing that Linux NFSv4 was not ready for use on 64bits platforms with kernel 2.6.12.

Now, starting with 2.6.15 kernel, all these issues have been fixed; and these interoperability tests are now included in our regression testing process.

	Client		
Server	ia32	x86_64	ppc64
ia32	OK	(1)	OK
x86_64	OK	OK	(2)
ppc64	(3)	(4)	(3)

(1) On x86_64 platforms, lot of socket resets.
(2) Input/output error: cannot close a big file.
(3) On ppc64 client, locking does not deliver the expected behavior.
(4) Socket: error -11 when doing write/read operations on 30 MB files.

Table 1: Interoperability testing result matrix on 2.6.12.

Software interoperability

Tests are run to detect problems between Linux implementation of NFSv4 and other Unix implementations. Since kernel 2.6.12, no problem has appeared about basic features of Linux NFSv4 (Client or Server) when interoperating with NFSv4 on Solaris 10 and AIX 5.3.

Future testing

Tests involving security have not been performed yet with different architectures and with other non-Linux Operating Systems. First tests to be run should use: Kerberos (krb, krb5i, krb5p) and SPKM.

4.2.6 WAN testing

NFSv4 has been designed to work over LAN[9] as well as over Internet. So we have run tests to stress Linux NFSv4 over WAN[10]. We used the same stress tools we used for LAN testing:

[9]Local Area Network.
[10]Wide Area Network.

Fsx, and Fsstress. But the testing process over WAN differs in the hardware environment.

Tests have been run between the CITI in Michigan and the labs of BULL SAS in Grenoble, France. Though these tests appeared to be very useful because several problem were found, setting up and configuring machines for tests was painful: patching and updating the kernel is not easy through Internet.

To help us, the NETEM tool has been deployed to emulate the behavior of Internet: high RTT[11] delay, high RTT variations and packets loss. This test environment has successfully helped us to find WAN-only bugs, like timeouts. Then, the new kernel containing appropriate fixes has been tested over the real Internet connection, between USA and France, and proved reliable.

Many problems have been fixed and WAN testing covers most core NFSv4 functions.

4.2.7 Future plans

Once NFSv4 is widely used, people will probably ask for proof that a NFSv4 server connected to the WAN cannot be used as a mean for penetrating a private network. After a first attempt in end of 2005 to analyze the Linux NFSv4 code by means of tools like Duma or Checker, we plan to start a more ambitious study: attack a Linux NFSv4 server with a modified Linux NFSv4 client in order to search for weaknesses in the Linux NFSv4 code, like overflows commonly used by attackers.

In the second half of 2006, we plan to continue NFSv4 regression testing, in order to continue finding problems in new kernel+NFSv4 versions as early as possible.

Running tests helps to improve the quality of the newest versions of NFSv4. It is also very important to deliver new test suites which will be used by Linux distributions for checking that NFSv4 behaves perfectly well in the ecosystem of their distribution. So we plan to continue writing tests for new features (like replication and migration, NFSv4 ACLs, named attributes) and to deliver them to the LTP project, as we did for ACLs and Locks tests in 2005.

[11]Round Trip Time.

4.3 Novell

NFSv4 support is available in SLES 10 by default. Novell started contributing to the testing efforts by taking part in OSDL Testing conference calls and providing inputs. Later we derived the test plan from OSDL test matrix and started testing.

All SLES 10 beta builds are being validated. The validation sequence is:

- Pynfs/NewPyNFS for protocol conformance,
- Connectathon 04 for basic conformance testing,
- Custom usability script which tests areas which are not covered by Connectathon,
- Support for security modes (sys, krb5, krb5i),
- Lock tests.

The major focus is on functionality, robustness, interoperability and performance. Protocol conformance, POSIX conformance, installability, integration testing, use case scenarios, and kerberos security modes were tested as part of functionality testing. Stress testing, comparison against NFSv3, various security modes, and various load scenarios were done as part of performance testing. Robustness Testing incorporates fsstress, ffsb, resource limit testing and crash recovery. The interoperability testing includes NetApp and Solaris platforms and all SLES 10 supported architectures.

Tools used

The tools used during various levels of testing are: PyNFS, OpenPOSIX, LTP, Connectathon, IOZone, fsstress, ffsb, Bull LockTester, and custom usability scripts.

Future plans

Our future focus will be tests like NFSv4 exporting cluster filesystem, NFSv4 testing under XEN kernel, scalability, and other pending items in performance and robustness testing.

4.4 Red Hat

Red Hat NFS developers contribute to two code streams, Red Hat Enterprise Linux and the community Fedora Core project. Red Hat Enterprise Linux has supported portions of NFSv4 for the last couple of years. Fedora Core closely tracks upstream development. By carefully monitoring bug reports against Fedora Core, we are able to gauge the maturity and readiness of the upstream bits for our enterprise customers.

Red Hat has an automated test harness which incorporates many of the tests mentioned earlier, among others:

- LTP
- fsx
- fsstress
- Locks tests
- ACL tests
- Connectathon suite

We run these tests against our nightly baselevels and compare the results against known good builds for regressions or problems. In addition, we have a performance group who run many different benchmarks with a variety of application mixes and closely monitor any unexpected performance changes (we scan for deltas of more than 3% to 5%).

We work closely with many key partners, who run their own test suites. In some cases, these test suites are proprietary, so when problem areas are encountered, we work together to come up with reproducible test cases which we can add to our test harness. We also work closely with targeted customers during beta to test specific new functionality in an effort to leverage unique customer expertise or environments. We monitor those results to see where we need to enhance our internal testing and code reviews.

Red Hat products are also fully deployed in production throughout the company, and development builds are deployed in test labs and other controlled environments. There are few quality metrics more powerful than having to face your angry co-workers on your way to the coffee pot!

Red Hat NFS developer Steve Dickson is a Connectathon regular along with other Red Hat employees. This is a very valuable opportunity to ensure maximum interoperability of both our stable RHEL builds as well as our latest development trees.

5 Conclusions

Status of NFSv4 protocol
Future of NFSv4

There is a variety of extensions to the NFSv4 protocol under development, and Linux is serving as an important testbed for all of them.

Directory delegations allow enhanced (read-only) caching of directory data.

pNFS allows NFSv4 clients to perform file IO using alternate methods, including parallel IO to multiple file servers and direct access to block storage.

The Sessions extension fixes some of the transport problems that have long plagued NFS, finally allowing a reliable replay cache that can ensure only-once semantics.

Status of Linux NFSv4

This paper has shown that the efforts to date have improved NFSv4 reliability and performance since 2004 up to now.

The early regression testing has helped developers to quickly isolate and fix defects. Tests done by OSDL and Bull have put NFSv4 into high pressure situations, in LAN and WAN modes. The NFSv4 test community's efforts have been successful in identifying many minor, medium and bad problems, on 32 and 64 bits architectures. Though the main testing activity is done on ia32, tests are also done on x86_64, PowerPC and ia64 processors. And everyone knows that shaking code on different architectures really helps in finding hidden mistakes and bugs!

Also, the availability of new test suites (ACLs and Locks tests) through the LTP helps Linux distributions to check that NFSv4 integrates perfectly in their specific ecosystem.

The comparison of NFSv4 with NFSv3 has shown the benefits delivered by NFSv4: high reliability and very good performance on TCP. Also, interoperability tests have shown that Linux NFSv4 nicely interoperates with other Unix NFSv4 implementations. People attracted by Samba will enjoy a new NFS version that delivers both Unix-Windows interoperability and very good performance when reading files, both in synchronous and asynchronous modes.

When we compare the status of Linux NFSv4 today against where it was when we started in late 2004, we can see that the testing efforts have generated significant improvements in all test areas and that the core of Linux NFSv4 is stable and powerful. Indeed, the NFSv4 infrastructure has attained quality standards which surpass NFSv3 in many cases and offer security levels that today's users are desperate for.

Now that Novell and Red Hat have started testing NFSv4 in depth on their distributions, this is a clear signal for companies and individuals that NFSv4 is ready to use on Linux. First for experimentations, and soon in the field, where Linux NFSv4 must prove that it scales nicely with hundreds or thousands of clients.

Future of NFSv4 testing

There is still much functionality in development for NFSv4, thus the Linux NFSv4 test project will continue. New tests will be written for testing the future Linux NFSv4 features: Named Attributes, NFSv4 ACLs, Replication, Migration. Also, the testing coverage must be measured to know the percentage of NFSv4 code that is exercised when running tests over NFSv4.

References

[1] IETF: RFC 3530: NFSv4 Protocol:
`http://www.ietf.org/rfc/rfc3530.txt`

[2] IETF: NFSv4:
`http://www.ietf.org/html.charters/nfsv4-charter.html`

[3] Paper: The NFS Version 4 Protocol:
`http://www.nluug.nl/events/sane2000/papers/pawlowski.pdf`

[4] Paper: Linux NFSv4: Implementation and Administration:
`http://lwn.net/2001/features/OLS/pdf/pdf/nfsv4_ols.pdf`

[5] NFSv4 ACLs :
`http://www.ietf.org/internet-drafts/draft-ietf-nfsv4-acls-00.txt`

[6] CITI: NFSv4 Open Source Reference Implementation:
`http://www.citi.umich.edu/projects/nfsv4/`

[7] CITI: NFSv4 for Linux 2.6 kernels:
`http://www.citi.umich.edu/projects/nfsv4/linux/`

[8] Connectathon:
`http://www.connectathon.org/`

[9] IOzone filesystem Benchmark:
`http://www.iozone.org/`

[10] LTP (FSX, ACL & Lock tests):
`http://ltp.sourceforge.net/`

[11] Ethereal (Network Protocol Analyzer):
`http://www.ethereal.com/`

[12] FFSB: Flexible File System Benchmark:
`http://sourceforge.net/projects/ffsb/`

[13] NetEm: Network Emulator:
`http://linux-net.osdl.org/index.php/Netem`

[14] NewPyNFS (NFSv4 Functionality Test):
`http://www.citi.umich.edu/projects/nfsv4/pynfs/`

[15] OSDL: NFSv4 Testing for Linux:
`http://developer.osdl.org/dev/nfsv4/site/index.php`

[16] OSDL NFSv4 Test Matrix v1.13:
`http://developer.osdl.org/dev/nfsv4/site/testmatrix/testmatrix-1.13.pdf`

[17] OSDL NFSv4 Wiki:
`http://wiki.linux-nfs.org/index.php/Main_Page`

[18] NFSv4 Bugzilla:
`http://bugzilla.linux-nfs.org/`

[19] Linux NFSv4 Client and Server Mailing Lists:
`http://linux-nfs.org/cgi-bin/mailman/listinfo/nfsv4`

[20] Linux NFS mailing list:
`http://lists.sourceforge.net/lists/listinfo/nfs`

[21] Bull: NFSv4 project:
`http://nfsv4.bullopensource.org/`

[22] Novell: SUSE Linux:
`www.novell.com/linux/suse/`

[23] Red Hat:
`http://www.redhat.com/`