Center for Information Technology Integration

NFSv4 Extensions for Performance and Interoperability

Final Report to EMC
February 15, 2007 – February 14, 2008

Background

When this project began, CITI's block layout client implementation was based on the Linux 2.6.17 kernel, NFSv4.1 draft 9, and pNFS block layout draft 1. Testing was done with a server on the Blackbird Celerra simulator, also based on Linux 2.6.17, NFSv4.1 draft 9, and pNFS block layout draft 2.

One of the goals in this phase of the work has been to keep current with the Linux kernel, which tends to see a new version every other month, the NFSv4.1 draft, which has been updated 10 times in the past year, and the pNFS block layout draft, which has undergone four revisions. Another goal has been to base our work on the generic pNFS client under development by the Linux pNFS group.† The tension between these goals complicated our work.

To simplify development and testing, the Linux pNFS group fixed its development of the generic pNFS client operations on the Linux 2.6.18.3 kernel for the first eight months of the project. Following the October 2007 Bakeathon in Ann Arbor, the group engaged in a community effort to rebase to the current Linux kernel. The effort was huge, as many of the NFS underpinnings had undergone significant change over the time span.

The following table shows the kernel and draft versions that were the basis of development in this project at the start, the testing events, and at the end, as well as the versions that were current at those times.

Event	Linux kernel	NFSv4.1 draft	pNFS block draft
Project start February 15, 2007	development: 2.6.17	development: 9	development: 1
	current: 2.6.20	current: 9	current: 1
Austin Bakeathon June 11–15, 2007	development: 2.6.18.3	development: 10	development: 3
	current: 2.6.23-rc2	current: 10	current: 3
Ann Arbor Bakeathon October 8–12, 2007	development: 2.6.18.3	development: 13	development: 3+
	current: 2.6.23-rc9	current: 14	current: 4
Project end February 14, 2008	development: 2.6.24	development: 13	development: 3+
	current: 2.6.25-rc1	current: 19	current: 5
Austin Bakeathon February 18–22, 2008	development: 2.6.24	development: 19	development: 3+
	current: 2.6.25-rc1	current: 19	current: 5

As the table suggests, we are on the cusp of rebasing to the current NFSv4.1 draft, in tandem with the rest of the Linux pNFS group, in preparation for the Austin Bakeathon.

The pNFS block draft used in development contains some elements of later drafts, but remains based on draft 3, in part to be compatible with XDR formats understood by the Celerra server.

† The Linux pNFS group—engineers from CITI, IBM, Panasas, Network Appliance, and EMC—works together to implement the NFSv4.1 and pNFS client and server in the Linux kernel.

Accomplishments

- CITI implemented basic pNFS block layout capability in the Linux NFSv4.1 client. This was the major goal of the project.
- As a member of the Linux pNFS group, CITI implemented sessions in the Linux NFSv4.1 client and server.
- CITI substantially completed rebasing the pNFS client block layout implementation to the current Linux kernel, NFSv4.1 draft, and pNFS block draft.
- CITI extended the Python-based PyNFS server to support generic pNFS operations.
- CITI implemented a PyNFS pNFS block layout server suitable for comprehensive testing.
- CITI installed and configured a Celerra system and is using that for its continuing development.
- CITI implemented directory delegation in the Linux client and server and rebased it to the current Linux kernel.

The remainder of this report enumerates the specific tasks in the project Statement of Work (*shown in italics*) and reports on their status in detail.

Task 1: pNFS

Task 1.1: I/O path

The I/O path in the current implementation goes directly to the block device cache, bypassing the NFS page cache. This introduces some size mismatch problems when writing partial blocks. CITI will change the I/O path to use the NFS page cache, which is already equipped to manage zero filling of partial block writes.

We earlier reported completion of this task for the Linux 2.6.18.3 kernel. The additional effort to rebase to the current Linux kernel was complicated by major changes in the underlying NFS page cache, but this task is now **complete** in the Linux 2.6.24-rc6 kernel.

Task 1.2: I/O path

The NFSv4.1 specification requires that errors in direct I/O fall back to conventional RPC requests to the NFS server. Task 1.1 changes the I/O path to use the NFS page cache, which is already equipped to issue conventional RPC requests. In this task, CITI will eliminate the complex error handling introduced to support an I/O path that went directly to the block device cache.

We eliminated the complex error handling, a vestige of the former I/O path that went directly to the block device cache. This task is now **complete** in the Linux 2.6.24-rc6 kernel.

Task 1.3: GETDEVICELIST, GETDEVICEINFO, and LAYOUTGET update

The recently revised block layout specification retrieves the device topology with OP_GETDEVICELIST instead of OP_LAYOUTGET. This requires changes to the existing block layout driver in CITI's client and EMC's server. In lieu of a server that understands the new XDR formats, CITI will use the PyNFS suite to test the modified driver.

To promote acceptance by Linux kernel maintainers, CITI will investigate the potential to reuse the topology management functionality in the existing Linux kernel multi-device driver (driver/md), which maps a file offset into a device and block number.

This task is **complete** in the Linux 2.6.24-rc6 kernel implementation using NFSv4.1 draft 13 and block layout draft 3. The use of the kernel device mapper (driver/md) was initially prototyped in the Linux 2.6.18.3 implementation, then ported forward.

CITI and EMC engineers jointly **submitted** a position statement‡ describing our block layout module implementation to the Linux File System and Storage Workshop and have been **invited** to attend and to **present** our work for peer review.

‡ William A. Adamson, Frederic Isaman, and Jason Glasgow, "Parallel NFS Block Layout Module for Linux," CITI Technical Report 08-1 (February 2008).

Task 1.4: Client layout cache

The generic (i.e., storage class independent) pNFS client implementation supports a per file whole file layout cache designed to serve file layouts. The block layout driver expands this simple cache to hold two layouts per file based on I/O mode. This is sufficient for simple I/O processing, but inadequate for complex block layouts. CITI will extend the layout cache to support complex topologies.

The object layout driver faces the same problem; CITI will investigate the potential to leverage our efforts by defining and implementing a common mechanism for caching complex layouts.

In the year that has passed since this task was proposed, changes in the pNFS block layout specification have obsoleted the task description and complicated the effort. The goal of this task is to provide a rich structure for block layouts and extents that supports complex topologies and copy-on-write. This task is **partially complete**.

An initial prototype in Linux 2.6.18.3 was obsoleted by changes in the block specification and by the generic layout cache developed by the Linux pNFS group.

We now use the generic client layout cache in the Linux 2.6.24-rc6 kernel. Late in 2007, we began making block-specific changes, e.g., merging compatible overlapping layout segments, but this code is not thoroughly tested, in part because the PyNFS-based server test rig uses whole file layouts. Copy-on-write functions with some restrictions. Further development and testing is required, both to the generic code and to our extent cache implementation.

Task 1.5: Client reboot recovery

Client recovery from server reboot follows the pattern established for recovery of other client state: a client reclaims its layouts after the server grace period ends. However, if the client is holding dirty blocks for a newly allocated extent, then the client should write that data (using NFSv4) during the grace period. In the event that the data no longer resides in the client cache, then the client has only one option: issue a LAYOUTCOMMIT with the reclaim flag during the grace period and hope that the pNFS server can rebuild the block layout from that. To guarantee freedom from data loss in this case, it is necessary for the client to maintain data in its cache until a LAYOUTCOMMIT succeeds. CITI will extend the layout cache to respond to server reboot.

This task is **incomplete** because it depends on session reboot recovery, which has not yet been implemented in the Linux 2.6.24-rc6 kernel.

Task 2: PyNFS block layout server suite

This task extends the PyNFS NFSv4.1 server to include block layout specific functionality. Without integration with iSCSI, the PyNFS NFSv4.1 server will not be able to perform real I/O, but it can test some important features of the client block layout driver.

CITI integrated iSCSI and RAM SCSI disks with the PyNFS NFSv4.1 server to provide a Python pNFS block layout server that can perform real I/O to a limited set of files. This **additional work** produced an NFSv4.1 PyNFS server (with pNFS file layout) that passes all Connectathon tests. Combined with the ability to export complex volume topologies by configuring the RAM SCSI disks via LVM2, the PyNFS server can test most of the block layout client implementation functionality.

Task 2.1

CITI will develop a suite of PyNFS NFSv4.1 pNFS block layout volume topology tests using OP_GETDEVICELIST and OP_GETDEVICEINFO. These tests will provide semantically correct volume topology variations to test the block layout client's ability to parse and manage complex topologies. This will require a "switch" on the pNFS block layout client to skip disk signature verification.

This task is **complete**: CITI developed a suite of PyNFS NFSv4.1 pNFS block layout volume topology tests using OP_GETDEVICELIST and OP_GETDEVICEINFO that exercise the block layout client's ability to parse and manage complex topologies.

Task 2.2

CITI will develop a suite of PyNFS NFSv4.1 pNFS block layout volume tests using OP_LAYOUTGET, CB_LAYOUTRECALL, and OP_LAYOUTRETURN to exercise the layout cache management in the pNFS block layout client.

This task is **complete**: CITI developed a suite of PyNFS NFSv4.1 pNFS block layout volume tests using OP_LAYOUTGET, CB_LAYOUTRECALL, and OP_LAYOUTRETURN that exercise layout cache management in the pNFS block layout client.

Task 2.3

CITI will develop a suite of PyNFS NFSv4.1 metadata server reboot recovery tests for the client block layout driver using OP_LAYOUTRETURN with the reclaim flag set.

This task is **complete**: CITI developed a suite of PyNFS NFSv4.1 metadata server reboot recovery tests for the client block layout driver using OP_LAYOUTRETURN with the reclaim flag set.

Task 3: Support latest draft

Full support for pNFS requires that CITI implement these portions of the latest NFSv4.1 minor version:

- *Support sessions operations EXCHANGE_ID, CREATE_SESSION, OP_SEQUENCE.*
- *Replace NFSv4.0 clientid and lock sequencing with sessionid and OP_SEQUENCE sequencing.*
- *Use OP_SEQUENCE sequencing (to prevent LAYOUTGET, LAYOUTRETURN races).*
- *Support the sessions callback infrastructure, and use it for CB_LAYOUTRECALL.*
- *Extend the pNFS client GETATTR processing to a 96-bit attribute bit mask and include all minor version pNFS attributes.*
- *Rebase the current block pNFS implementation (client and server) to the latest Linux kernel.*

This task was complete for the Linux 2.6.18.3 implementation subsequent to rebasing the block layout module from the Linux 2.6.17 kernel. CITI then worked with the Linux pNFS group to port forward to a git tree backed by the Linus git tree, which allows us to track new kernels. The forward port required CITI to rewrite the client and server sessions code and the pNFS client I/O path. That effort is now **complete**.

CITI rebased the pNFS block layout implementation to the Linux 2.6.24-rc6 kernel and is working with the Linux pNFS group to move to NFSv4.1 draft 19. Although CITI's pNFS block layout client is based on draft 3, it incorporates many of the changes to the draft 5 specification. CITI plans to rebase to draft 5 **within weeks**.

Throughout this project, CITI has **participated** in the refinement of the NFSv4.1 draft specification (567 pages!) and the pNFS block layout draft specification; this has been a considerable effort.

Task 4: Directory delegations

NFSv4 clients cache directory contents in the following ways:

- *READDIR uses a directory entry cache*
- *LOOKUP uses the name cache*
- *ACCESS and GETATTR use a directory metadata cache.*

To limit the use of stale cached information, RFC 3530 suggests the NFSv3 time-bounded consistency model, which forces a client to revalidate cached directory information periodically. This revalidation of directory information is wasteful — CITI's analysis of network traces reveals that a surprising amount of NFSv3 traffic is due to GETATTRs triggered by client directory cache revalidation. The NFSv3 model also opens a window during which a client might use stale cached directory information.

To improve performance and reliability, NFSv4.1 introduces read-only directory delegations, a protocol extension that allows consistent caching of directory contents. CITI has begun implementing directory delegations, as described in Section 11 of NFSv4.1 Internet Draft. A modest effort will move that work to completion. Beyond the performance advantages, completing the implementation directory delegations will help move the NFSv4.1 minor extension standardization forward.

This task is **complete**: directory delegations patches were ported the Linux 2.6.20 kernel, then to the Linux 2.6.24-rc6 kernel, and are now integrated into the Linux pNFS group NFSv4.1 git tree.