Center for Information Technology Integration
Proposal to EMC

NFSv4 extensions for performance and interoperability

February 15, 2006 – February 14, 2007

Statement of work

This is a statement of work to be performed by CITI in a project sponsored by EMC. The major goals of this project are:

- Prototype emerging NFSv4.1 features to evaluate performance,
- Develop NFSv4 features and auxiliary database usage to enhance interoperability in a multi-protocol server environment.

Task 1: NFSv4.1 pNFS extension

pNFS, a new feature for a future NFSv4 minor version, is in the middle stages of definition by the IETF. pNFS has great potential to improve NFS performance. In one scenario, a client can access a file in parallel through simultaneous I/O to a collection of storage servers.

The draft specification for pNFS [pNFS] specifies storage-class independent extensions to NFSv4 and direct I/O to file-based storage. The draft specification for pNFS block-based access [pNFS-block] specifies pNFS operations for block and volume based storage.

EMC's Celerra Highroad file system is similar in spirit to the pNFS block-based architecture and serves as a reference for the implementation of the pNFS block layout driver.

Prototype work has begun on Linux pNFS client extensions intended to provide full pNFS functionality by supporting pluggable storage-class specific extensions (called *layout drivers*) and a storage-class independent pNFS layer. The current storage class-independent pNFS client prototype addresses the needs of file and pVFS2 layout drivers. A block-layout driver is likely to require a more complicated layout driver interface.

Task 1
Implement a prototype block layout driver capable of operating on a single file in parallel using block I/O. The prototype will implement storage class-independent pNFS extensions and layout management API following specifications from the emerging draft specifications of pNFS protocol operations and block based pNFS.

A further breakdown of subtask 1.1 with milestones is shown below. Throughout the performance of Task 1, CITI will provide feedback to the authors, editors, and working group members of emerging pNFS drafts on experiences in addressing issues with data storage security, access, and client fencing.

All source code developed by CITI in this task will be freely available open source. CITI will work with Linux kernel developers and maintainers to facilitate acceptance into the mainstream Linux kernel.

[pNFS] G. Goodson, B. Welch, B. Halevy, D. Black, and A. Adamson, “NFSv4 pNFS Extensions,” draft-ietf-nfsv4-pnfs-00.txt (October 7, 2005).

[pNFS-block] David L. Black and Stephen Fridella, “pNFS Block/Volume Layout,” draft-black-pnfs-block-02.txt (October 21, 2005).

We can break down Task 1 into the following milestones.

T1 Month 1: Background

Work begins by studying the EMC File Mapping protocol, Celerra HighRoad implementation, Linux 2.6.15+ SCSI driver interface.

- Install EMC pNFS block-based file service.
- Test NFSv4.0 functionality via NFSv4.0 client to EMC pNFS MDS.

T1 Month 3: GETDEVICELIST and GETDEVICEINFO

A device ID may need to carry a lot of information, so GETDEVICELIST returns a list of device ID's, and GETDEVICEINFO needs to be called on each ID.

- “Ping” each device to confirm connectivity.
- Respond to changing devices.

T1 Month 5: LAYOUTGET, basic read/write

We anticipate some complications involving GETLAYOUT request and retry semantics and multiple layouts (read and read/write layouts) that cover the same range.

- Perform layout processing. Check pNFS layout iomode extent data type, extent coverage of min/desired size.
- Fetch and store data through the read/write interface.

T1 Month 7: LAYOUTCOMMIT

- Deal with “holes” in layouts. The four types of layout data — RWdata, Rdata, Invalid-data, and none-data — complicate LAYOUTCOMMIT processing and treatment of holes.
- Investigate pNFS client participation in copy-on-write.

T1 Month 9: CB_LAYOUTRECALL and LAYOUTRETURN

There is no layout StateID. NFSv4.1 sessions may be necessary here.

- Avoid races in LAYOUTGET/LAYOUTRETURN

T1 Month 10: CB_SIZECHANGED

- EOF processing.

T1 Month 12: Reboot recovery and testing

T1 Demonstration

At the end of the project, CITI will demonstrate the pluggable block-layout driver prototype co-existing with other pluggable prototype layout drivers (file, pVFS2) in the Linux pNFS client.

Notes

- *This task depends on EMC providing CITI with a block-based pNFS service and access to EMC pNFS server implementers.*
- *Any code provided by EMC to CITI will be open source but CITI will exercise reasonable care in restricting its distribution.*

Task 2: NFSv4 ACL

The NFSv4 protocol introduced access control to file-system objects through Access Control Lists (ACLs). RFC 3530 specifies the format and meaning of NFSv4 but leaves open some questions about semantics, e.g., for ACLs derived from mode-bits, order of the ACEs in the ACL, etc, which may lead to interoperability problems, lots of grief to the end-user, large number of support calls, etc. Some of the problems we encountered so far are:

- NFSv4 servers translate the mode-bits into an ACL that is not understood by all NFSv4 clients.
- An ACL set by one client may not be properly understood by another type of client (example: Solaris vs. Linux). More and more customers use clients on different platforms. Having the same "look and feel" from different clients is important.
- Lack of interoperability with the Windows environment. Some servers serve NFS and SMB environments and the interoperability between these worlds is essential to viability of NFSv4. These servers have only one ACL per file-system object, independent of the protocol used to access the file-system.

Note that the solution may include some translation of ACLs on the CIFS/NFSv4 multi-protocol server.

Task 2
Build consensus around solutions to NFSv4 ACL interoperability problems, build a Linux client, demonstrate and document its interoperability with a windows CIFS client and a multi-protocol server.

Task 2 breaks down into the following steps and milestones.

T2 Month 2: White paper
• Prepare a white paper that details NFSv4 ACL interoperability problems candidate and proposes solutions; circulate to EMC and the NFSv4 community for commentary.

T2 Month 4: Internet draft
• Publish or contribute to an Internet draft that details NFSv4 ACL interoperability problems and proposes consensus solutions. The solution may involve ACL translation on the multi-protocol server. • Prototype the Linux NFSv4 client according to the proposed specification.

T2 Month 6: Build, test, iterate
• Refine the NFSv4 ACL interoperability Internet. • Prototype the Linux NFSv4 client according to the proposed specification. • Test setting and retrieving ACLs using the Linux NFSv4 client prototype and a windows CIFS client against the multi-protocol server.

T2 Month 8: Demonstrate ACL interoperability

Notes

- *Task 2 depends on EMC providing CITI with a multi-protocol CIFS/NFSv4 service for ACL testing.*

Task 3: Global name space

An NFSv4 server controls the name space it exports by grafting together volumes. An NFSv4 client that encounters a graft is informed with NFS4_ERR_MOVED. The client then examines the FS_LOCATIONS attribute to locate the target volume.

The FS_LOCATIONS attribute provides the client with a list of referrals — servers where the target volume can be located. Referrals are useful for migration and replication, as well as basic name space construction.

This task involves the construction, management, and retrieval of the FS_LOCATIONS server lists for heterogeneous NFSv4 implementations, with the intention of utilizing the Active Directory schema used by Microsoft DFS junctions. If DFS junction schema can be used, then Windows clients and NFSv4 clients can see the same namespace via a common back-end management.

Task 3
This task involves basic NFSv4 name space construction and interoperability with Windows DFS. Build consensus around a common LDAP/DFS AD schema for FS_LOCATIONS information, demonstrate and document Linux client. CITI will work with Linux kernel developers and maintainers to facilitate acceptance of this effort into the mainstream Linux kernel.

Task 3 breaks down into the following milestones.

T3 Month 1: OpenLDAP schema
• Complete the Linux NFSv4 client and server support for FS_LOCATIONS. • Design an OpenLDAP directory service schema for FS_LOCATIONS information and prepare a white paper for circulation to • Prepare a white paper that describes an OpenLDAP directory service schema for FS_LOCATIONS information; circulate to EMC and the NFSv4 community for commentary.

T3 Month 2: Investigate AD
• Investigate the use of DFS Active Directory junction schema for FS_LOCATIONS information.

T3 Month 3: Demonstrate client interoperability
• Demonstrate interoperability of the Linux NFSv4 client with the Linux NFSv4 server and EMC server using the same OpenLDAP schema. (This may involve adding to Active Directory junction schema.)

T3 Month 6: Demonstrate client and server interoperability
• Demonstrate a Windows DFS client and a Linux NFSv4 client viewing the same file namespace with an EMC server and an Active Directory back-end.

Notes

- *This task depends on EMC providing CITI with a multi-protocol CIFS/NFSv4 service for namespace construction testing.*

Resources

Task 1 (NFSv4.1 pNFS extension): Twelve FTE-months in twelve months.

Task 2 (NFSv4 ACL): Four FTE-months in eight months.

Task 3 (Global name space) Two FTE-months in six months.

All tasks will start on the first day of the project.