CITI - ARC Joint Project Statement of Work

This is a proposal for a joint project between IBM, Almaden and CITI, University of Michigan. The goals of this project are:

- Enhance the Linux NFSv4 client and server to operate correctly with a cluster file system
- Exploit emerging NFSv4 features to improve performance, fault-tolerance, and load-balancing

The goal of this development effort is open source software released through appropriate Linux maintainers. From IBM's point of view, the file system of choice will be GPFS, but the development should involve interaction with the open source code community and other cluster file system maintainers to define interfaces that are acceptable to all.

Task 1: Locking, Delegations, and Shares for Cluster File systems

Correct operation of an NFSv4 server and a cluster file system with multiple servers requires some file operations to be synchronized with the cluster file system. Specifically, byte range locks, open share modes, and delegations must be passed to the file system through extensions to the VFS interface (or other means).

This task involves the definition and implementation of the following:

- Uniform interfaces for different lock managers (POSIX, lockd, nfsd) that invoke file system-provided VFS lock(). (This may require a new lock() interface that returns the conflicting lock). This involves defining a fair-queue scheme for handling blocking requests and providing a callback interface for reporting lock availability.
- New interface for open share modes that need to be supplied with NFSv4 OPEN call.
- New interface for acquiring file delegations before they are granted to the clients including callbacks for revocation.

Task 2: Replication and Migration

NFSv4 has a means of providing file system replication and migration services through the recommended FS_LOCATIONS attribute. Allowing clients to move seamlessly from one NFSv4 server to another provides fault tolerance through replication, load balancing, migration in a global namespace, and flexible resource allocation. While replication is intended for read-only data, certain file systems like GPFS that provide a cluster solution can exploit this feature for read-write data as well. This task involves complete implementation of the optional features of RFC 3530 that relate to replication and migration.

This includes the following:

- Detection and handling of replication and migration events originating from an NFSv4 server at the client. This includes support for volatile file handles, file handle recovery on expiration, following of referrals and state recovery or expiration on migration.
- Server support for FS_LOCATIONS and generation of migration and referral events on a trigger, possibly the exports file. This trigger can help a performance monitor to do load balancing, and an administrator to move NFSv4 clients between NFSv4 servers that have a common backend.
- Definition of a server interface for obtaining FS_LOCATIONS information for a given file system, either from an external source (e.g., LDAP database) or through the file system.

Task 3: pNFS

pNFS, a new feature for a future NFSv4 minor version, can improve performance through parallel access to storage servers. We would like to experiment with an implementation and by doing so, help in the definition of this emerging protocol. Although the plan is for pNFS to support object servers and block devices, we would like to focus on the multiple NFS servers version. Parallel file I/O involves the client operating on one file by concurrently doing I/O to multiple NFSv4

servers.

This task involves the following:
- Extending the NFSv4 client to operate on a single file in parallel using specifications from the pNFS protocol extension
- Experimentation with FS_LOCATIONS as file attributes for obtaining multiple locations for a file.

Task 4: State Migration

The goal of this task is to explore options in state migration. Section 8.14 of RFC 3530 states that servers involved in migration SHOULD transfer all server state from the original to the new server in a way that is transparent to the client. For cluster file systems that support persistent file handles, this case applies when the client has accessed a file system that is now moved to another server. (In a pure referral, state does not have to be transferred, but can expire). For cluster file systems to do effective load balancing using the migration capabilities in NFSv4 (as opposed to referrals), state MUST be migrated. It would be useful to support state migration in the server; whether the file system should be involved in the transfer is open to debate.

Goals

All the tasks listed above involve the following:
- Jointly document a detailed design for the above components.
- In general, CITI will lead implementation efforts in Linux kernel. Any file system extensions required in GPFS will be done by Almaden. Almaden will also provide prototypes of file locking and NFSv4 migration and replication support. Almaden will have the responsibility of testing all implementations with GPFS; other (open-source) cluster file systems will be tested by CITI if required. We believe testing with an open source file system is crucial for acceptability into the kernel mainline.
- Successful completion of tasks 1-3 would be acceptance into the kernel mainline. Tasks 4-5 involve developing prototype implementations and possible involvement in Internet-draft specification. It is also important that this project be in harmony with other NFSv4 activity (in particular, the PolyServe and GFS work) so that the result is a coherent solution.

Schedule

This project will span twelve months, with possible extensions if needed. The first component (locking) should be tested and merged into the kernel mainline in three months. The second component (FS-LOCATIONS) should be demonstrated after six months and merged into the kernel mainline within nine months. The third component (pNFS) should be demonstrated after nine month and merged into the kernel mainline code upon final definition of the pNFS protocol.