CITI - ARC Joint Project Statement of Work
October 2006 – September 2007

This is a proposal to extend a joint project between IBM Almaden Research Center (ARC) and the University of Michigan Center for Information Technology Integration (CITI). The goals of this project remain to enhance the Linux NFSv4 client and server to operate correctly with a cluster file system, and to exploit emerging NFSv4 features to improve performance, fault-tolerance, and load balancing.

The primary deliverable of this project is open source software, released through appropriate Linux maintainers. While IBM's file system of choice is GPFS, the project strategy is to define generic interfaces that are broadly accepted by the open source community. To meet the needs of other cluster file system stakeholders, the development process welcomes interaction.

1 Locking, delegations, and shares

Correct operation of a cluster file system with multiple NFSv4 servers requires that byte range locks, open share modes, and delegations be synchronized by the cluster file system.

Byte-range locks

Together, CITI, ARC, and other stakeholders have made good progress on byte-range locks by coalescing the locking requirements of NFSD, LOCKD, and the Linux local lock manager into a single interface and a single queue. To provide for fair queuing, we extended the lock interface to issue “provisional” locks on behalf of waiting lock contenders. We are still in the process of gathering consensus among Linux maintainers in accepting these extensions, but with GFS now part of the mainline Linux kernel, the requirement for a solution to the fair queuing problem takes on elevated importance.

Task 1.1
• Continue to develop and test fair queuing for byte-range locks. • Respond to feedback from Linux developers and maintainers. • Work with Linux maintainers to include a solution for fair queuing in the mainline kernel.

Open share modes

While NFSv4 ACCEPT READ, ACCEPT WRITE, ACCEPT BOTH, and DENY NONE open modes map directly onto the Linux POSIX interface, an atomic implementation of DENY READ, DENY WRITE, and DENY BOTH is not so straightforward. To date, CITI and ARC have wrestled with a range of options, but the problem resists an easy solution.

Task 1.2
• Design and implement an interface for NFSv4 open share modes.

Delegations

CITI and ARC have proposed an interface for managing delegations in cluster file systems that extends the set/get/break lease interface, but implementation, test, and acceptance remain.

Task 1.3
• Implement and test the proposed cluster-friendly interface for file delegations.

2 Replication and migration

Support for client migration opens the door to fault tolerance through replication, load balancing, migration in a global namespace, and flexible resource allocation. The NFSv4 protocol uses the (recommended) FS_LOCATIONS attribute to provide seamless client migration from one NFSv4 server to another. While NFSv4 replication is designed with read-only data in mind, cluster file systems can exploit migration for read-write data as well.

This task entails the design and implementation of a mechanism for server state migration and a complete implementation of the NFSv4 replication and migration features.

State migration

In prior work, CITI and ARC concluded that a pure referral approach to client migration introduces unacceptable latencies to clients of the target server, which mandates transferring some state from the initial server to the target server. To prepare for state transfer, we redesigned server state bookkeeping to ensure that state transferred from one server will not collide with state created on another server. We then recoded the server interface used to transfer recovery state to nonvolatile memory to use a `pipefs` interface instead of VFS. Next, we expanded that interface to allow it to pass migration state.

Task 2.1
• Implement and test applications that support server-to-server state transfer.

The next step is to inform migrating clients that state established with the initial server is valid on the target server. The IETF NFSv4 working group is considering solutions for the NFSv4.1 protocol, but NFSv4.0 clients will not have support for this feature, so we need a Linux-specific implementation, perhaps using a mount option or a /proc flag.

Task 2.2
• Design and implement a client notification mechanism to inform migrating clients that state established with the initial server is valid on the target server.

Client migration

The Linux NFSv4 client supports referrals and (read-only) replication by selecting a server on the referral list and reissuing the request. Migration is complicated by the need to reestablish client state on the target server. Once state migration is in prototype, we can press forward with migration and replication.

Task 2.3
• Design and implement client support for volatile file handles, file handle recovery on expiration, following of referrals, and state recovery or expiration on migration. • Design and implement triggered generation of server migration and referral events.

The Linux NFSv4 client uses /etc/exports directly or LDAP for referrals and (read-only) replication. We will use the same mechanism for migration. We made good progress in organizing client state associated with an FSID to help state management.

Task 2.4
• Implement and test a server interface to FS_LOCATIONS information for a given file system that uses the exports file and LDAP.

3 pNFS

pNFS development and testing continue to show promise. Sustained effort is needed to promote IETF standardization and mainline Linux kernel inclusion.

Task 3.1
• Continue to implement and test NFSv4.1 features, including sessions, RECLAIM_COMPLETE, the generic pNFS client and server, and the file layout driver. • Respond to IETF and Linux kernel developer comments

Coordination

Success is understood to be acceptance of solutions into the mainline Linux kernel. It is also important that this project be in harmony with the efforts of other stakeholders, e.g., PolyServe, GFS, and Google.

CITI will lead Linux kernel implementation efforts. CITI is responsible for testing with other open source cluster file systems, a key step toward acceptability into the mainline Linux kernel. CITI is responsible for coordinating extensions specific to GFS2 and for coordinating comprehensive GFS2 testing. ARC is responsible for any extensions specific to GPFS and for comprehensive GPFS testing.

This project will span twelve months, beginning October 1, 2006, with possible extensions if needed. The primary channel for reporting will be weekly conference calls. CITI will submit a comprehensive report to ARC no later than November 30, 2007.