Center for Information Technology Integration

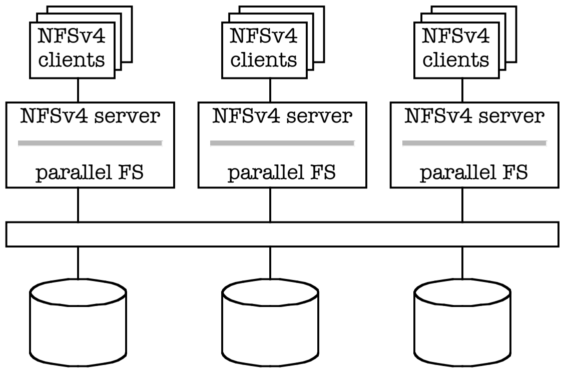
NFSv4 for Parallel File Systems

Statement of work

CITI proposes to extend its Linux NFSv4 reference implementation to meet the mutual objectives of Polyserve and the University of Michigan. The following tasks result from discussions between CITI and Polyserve technical staff.

Task 1: Enable symmetric NFSv4 servers on a single parallel file system

The goal of this task is to export the parallelism of an underlying file system through independent parallel NFSv4 servers.



Here, the NFSv4 servers see an identical underlying parallel file system. In this task, we make some simplifying assumptions:

- The underlying parallel file system provides data and lock coherency guarantees.
- An NFSv4 client mounts a single NFSv4 server.
- We are not concerned with replication, migration, or failover.

However, NFSv4 locks across multiple servers must provide appropriate guarantees to clients.

Our proposed approach to this task is to identify the appropriate VFS calls necessary for supporting a parallel file system, implement and test them, work with others in the Linux community to refine the architecture, and make appropriate implementation changes. The resulting prototype is the principal deliverable for this task.

To identify the appropriate VFS calls, we will analyze NFSv4 server state and determine those state elements that require support from the underlying file system. Our preliminary analysis indicates VFS support is needed to set and get two elements:

- share/deny rights, and
- delegation state.

CITI will implement the VFS extensions and test them with a freely available parallel file system such as PVFS2. At the same time, Polyserve will implement the extensions to its file system.

With a candidate architecture in hand, CITI will reach out to other stakeholders in the Linux parallel file system space. Their feedback will be reflected in changes to the architecture and implementation.

Task 2: Client migration and server failure recovery

In this task, we will address the problem of moving the responsibility for serving a file system from one NFSv4 server to another. As a tactical matter, we propose to use the standard NFSv4 server reboot recovery and the client state reclaim mechanism to expedite the implementation. In combination, these will provide reboot recovery, file system migration, client migration, and failover.

Anticipating a delay in universal commercial support for FS_LOCATIONS, we propose to implement this feature emulating Polyserve's NFS migration mechanism that transfers the IP address for a logical NFS server from one physical server to another. However, CITI's approach will work with FS_LOCATIONS as well, should that be available and desirable.

The first subtask is to complete the implementation of reboot recovery on the Linux NFSv4 server, as specified in RFC 3530, namely storing ClientID state and associated lease information to stable storage and retrieving it on reboot. This allows a server to recognize valid reclaim requests, i.e., ClientIDs that it had been serving.

We need to invent a mechanism that allows us to "hand off" the valid ClientIDs from one server to another. We favor using the underlying shared file system.

We will use the hand-off mechanism to migrate a client. Polyserve's current approach moves the IP address of a logical NFSv4 server from one physical server to another. We will support that.

With a working hand-off mechanism, the client state reclaim and server reboot recovery mechanisms do the rest. Moving a group of clients, e.g., those associated with a logical IP address, is straightforward, given the ability to move a single client. For failover, we will implement a bookkeeping procedure that associates client state with NFSv4 server (logical) IP addresses. When a server fails, the ClientIDs in stable storage can be handed to a working server.

Implicit in this task is a conversation between CITI, Polyserve, the Linux NFS server maintainer, and other stakeholders to achieve consensus on flexible mechanism for specifying the location of stable storage.

The approach outlined here requires no NFSv4 protocol extensions, allows (but does not depend on) FS_LOCATIONS, and allows (but does not require) the Polyserve strategy of virtualizing the server IP address for failover. We believe that this approach will make migration transparent to applications running on NFSv4 clients, obviating any need to suppress I/O failure notifications.

Task 3: Consistent file handles

File handles in NFSv4 are generally constructed from device-specific information, such as major and minor device numbers. In a parallel file system, this information may be different on different servers for a given file. This interferes with the ability to use a file handle issued on one server to reference a file via another server, as the latter server may use different device numbers.

This issue is under discussion by Linux kernel implementers – see

http://cgi.cse.unsw.edu.au/~neilb/
NFSserver/01084406486

In this task, CITI and Polyserve will participate in the discussion. CITI will prototype viable solutions and work with Linux kernel maintainers to promote a solution compatible with Polyserve's requirements.

Task 4: Administrative tools for client migration

In this task, we will build tools for managing client/server associations.

One tool will construct FS_LOCATION referrals to construct name spaces and manage client migration. We favor a solution that provides fine-grained management of referrals, at the level of a single client and a single file system (FSID).

Another tool will cause a client to migrate from one server to another.

There are many options to consider; discussions with Polyserve and the Linux community will guide the design of the interface and implementation.

This task depends on the completion of FS_LOCATIONS and the way they are implemented, as well as on joint decisions made by CITI and Polyserve.

Task 5: Server-side named attributes

In this task, we will complete the server-side implementation of named attributes, as specified in RFC 3530. The implementation will be independent of Linux inode structures and will allow the registration of a named attribute interface on a per-file system or per-file system type basis.

Task 6: Test bed construction and maintenance

To accomplish the tasks outlined above, CITI will need to build a test bed that is a subset of the figure above. CITI will provide NFSv4 clients drawn from its existing pool of hardware resources.

CITI proposes that Polyserve equip CITI with two nodes for running NFSv4 over a parallel file system (subject to discussion, but PVFS2 stands out for a number of reasons). CITI will also need a storage back end for the parallel file system to serve.

Further discussions may determine that CITI also needs a working Polyserve installation for testing. CITI proposes to identify the hardware needs of the project in continuing discussions with Polyserve and that Polyserve lend the necessary equipment to CITI or provide funds for its acquisition.

From experience with similar projects, we can say with some assurance that the hardware test bed will require significant attention by a skilled technologist.