State Abstraction Layer (SAL)

Design Document

Author : Philippe Deniel (philippe.deniel@cea.fr)

Version 0.1

Abstract : this document presents the design proposal for SAL, a new abstraction layer to be introduced in nfs-ganesha and dedicated to state management

1 Why a new Layer ?

NFS-GANESHA has started its life as a NFSv3 daemon, so it primary was a merely stateless daemon. As its supports new protocols like NFSv4, NFSv4.1 and NLMv4 (ancillary protocol designed to managed locks with NFSv3), state management becomes necessary and should be considered in the design.

For the moment, state management has been implemented in several modules : FSAL has lock-dedicated calls, Cache_Inode manages state (locks and share reservations), NFSv4 and NLMv4 implementation have state-related stuff. The trouble is that each state based implementation is independent from the others where they all should be working together. This is a bad situation that is to be fixed in a short term delay. The intention with SAL is to design a new layer that will provide NFS-GANESHA with a way to manage state from a central point, making state management more homogeneous.

2 Aspects to be addressed by SAL

The SAL should be called and used each time states are to be managed. This covers the following cases:

- NFSv3 locks, covered by NLMv4 implementation
- NFSv4 share reservations
- NFSv4/NFSv4.1 locks
- NFSv4/NFSv4.1 file delegations
- NFSv4.1 directory delegations
- NFSv4.1/pNFS layouts

The SAL should be responsible for all the recovery process required by the state management and described in the NFSv4.x RFCs. These recovery API will be called by upper modules of nfs-ganesha as the daemon reboots.

3 Who will require the SAL's service ?

NFSv4.x, NLMv4 and pNFS are the reason why such a module has become a requirement. The logic in nfs-ganesha is to keep as much information in a metadata cache, the Cache_Inode layer. This modules wraps calls to the FSAL and keeps most of the FSAL's objects information in memory. The “final” operation done will be a FSAL operation most of the time : locking/unlocking a file, querying the FS about the “striped topology” of a file on a distributed FS. The rule is to prevent all protocols implementation to use FSAL, and use Cache_Inode calls instead. Because of this, Cache_Inode will be the one and only client to SAL.

On the other hand, SAL should be able to handle as well NFSv4.x and NLMv4, which means that its semantic should be “polymorphic” enough to fit the requirements for all protocols.