Secure Identity Management in Lustre 2.X

Joshua Walgenbach

April 24, 2012

INDIANA UNIVERSITY

Lustre WAN

Since 2008 IU's production Lustre WAN file system (DC-WAN) has let people access, manage, and share data in a simple and familiar way.

- The file system is the user interface for mounting locations
- Provides reliable data transfer
- Run applications across the net without transfer of voluminous data sets

In order to accomplish this, it has been necessary to span heterogeneous name spaces.

History

Spanning heterogeneous namespaces was first made possible for Lustre 1.4.X through a kernel module and patch developed by IU and DDN.

IU implemented subsequent versions for 1.6.X and 1.8.X.

Currently in use at IU as part of DC-WAN and as part of XSEDE's Lustre-WAN project, Albedo.

For widest adoption code alterations were made solely to the MDS – unfortunately this breaks quotas.

Structural changes to 2.X will require a rewrite, offering opportunities to right wrongs, extend the feature set, and simplify the management interface

OpenSFS

The board of OpenSFS approved funding to make the rewrite possible.

The rewrite will consist of two distinct and interrelated projects:

1. Static UID mapping which will provide flexible administrative control of heterogeneous namespaces.

2. Extensions to GSSAPI which will provide for shared key machine authentication and encryption.

UID Mapping in DC-WAN

- UID mapping is encapsulated in its own kernel module.
- Mapping happens only on the MDS.
- Maps are managed in user space via a /proc interface.
- Mapping applies to TCP networks only.
- Maps are indexed by NID ranges.
- Does not support quotas.

New UID Mapping Model

- Changes to the Lustre code base will be restricted.
- UID Mapping will still be encapsulated in its own kernel module.
- UID Mapping will no longer be limited to the MDS.
- Will provide quota support.
- MGS will hold the canonical map and update the other servers.
- Update method will borrow heavily from imperative recovery.
- Will organize client sets as clusters.

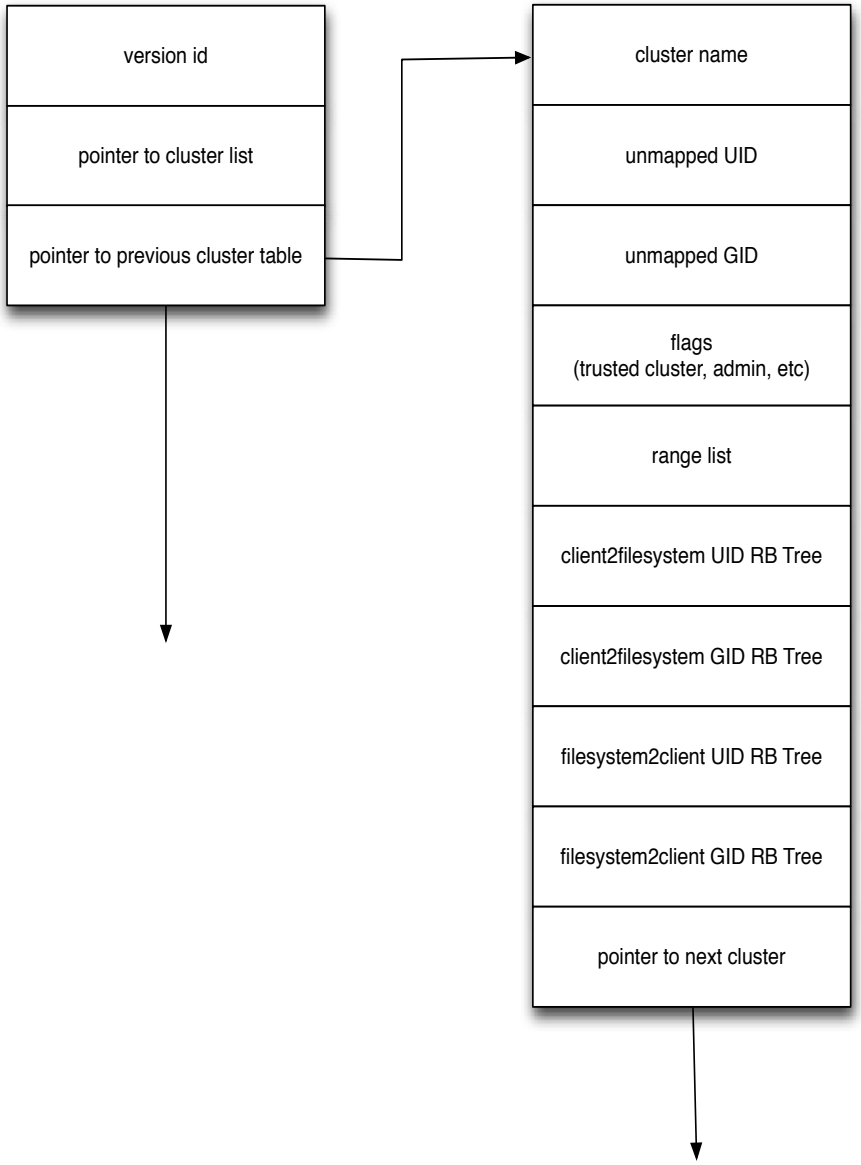
Clusters

We define a cluster as a partitioned set of NIDs (clients) that share a UID/GID name space.

It is a convenient way to think and talk about client nodes mounting the file system. It is also a convenient way to codify identity mapping.

When a client connects to a server, part of the process will be categorizing the client into a cluster, and thus giving it a pointer into the maps for forward and reverse UID/GID mapping.

This provides for a relatively small amount of memory usage when compared to a current (2.X) idmap implementation with the same mappings.


version id
pointer to cluster list
pointer to previous cluster table
cluster name
unmapped UID
unmapped GID
flags
(trusted cluster, admin, etc)
range list
client2filesystem UID RB Tree
client2filesystem GID RB Tree
filesystem2client UID RB Tree
filesystem2client GID RB Tree
pointer to next cluster

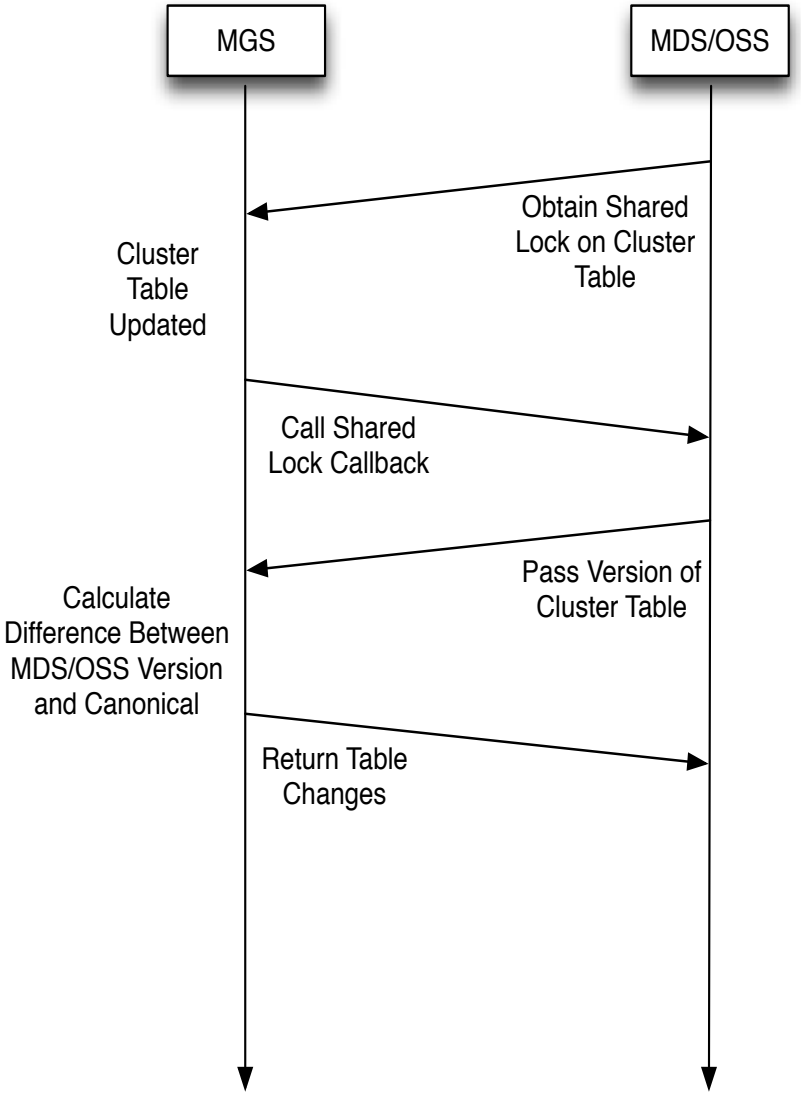
Updating Method

The Management server manages the cluster table.

When the cluster table is updated on the MGS, the MGS uses the callback provided in the request for the shared lock to inform the server that there is an update.

The server (MDS/OSS) then provides the MGS with the version id of the cluster table it currently holds in memory. The MGS calculates the delta between that version and the current canonical version, and provides the changes to be applied to the server (MDS/OSS).

This borrows heavily from the design of imperative recovery design.


MGS
MDS/OSS
Obtain Shared Lock on Cluster Table
Cluster Table Updated
Call Shared Lock Callback
Pass Version of Cluster Table
Calculate Difference Between MDS/OSS Version and Canonical
Return Table Changes

Trust

- The file system servers must trust the clients.
- No security scheme can protect a client from itself.
- The best we can do is mitigate the risk, and protect one client (or set of clients) from another.

Extensions to GSSAPI

Generic Security Services Application Program Interface

The security model in Lustre 2.X is properly called GSSAPI with a Kerberos mechanism.

GSSAPI in Lustre can be viewed as a vacuum cleaner with only one attachment.

We don’t want to ditch the vacuum cleaner, just add another attachment!

Existing GSSAPI Mechanism

- Not every organization runs (or wants to run) Kerberos.
- Not every organization wants to do cross-realm.
- Kerberos works best in an interactive environment.
  - User credentials time-out.
  - User credential distribution can be a challenge.
  - User credential keytabs are the equivalent of having the user store their password in a file.
- All GSSAPI security mechanisms are vulnerable to root compromise.
- Ultimately, user authentication only provides UID mapping.

Machine to Machine Authentication

GSSAPI provides the following functionality:

- Authenticating connections between machines.
- Authenticating machines passing messages to each other – Hashed Message Authentication Codes (HMAC) to prevent messages from being injected by a third party.
- Protecting the data on the wire by encryption.

Shared Keys

GSSAPI supports both HMAC and encryption. A backend mechanism needs to be written.

A shared key implementation can provide both HMAC and symmetric encryption without requiring a centralized infrastructure, for both RPC and bulk data transfer methods.

The Linux kernel crypto module hashing and encryption functions will be used – no need for reinventing the wheel.

Conclusion

Indiana University's development of a lightweight UID mapping scheme has provided researchers with the ability to share data across namespace borders within and outside the borders of institutions.

With OpenSFS support we will be able to extend and enhance previous work for wider adoption of Lustre across unsecured networks.

When development is complete, future versions of Lustre will provide static UID mapping and a non-centralized GSSAPI extension, paving a struggle-free path for Lustre administrators to help their users collaborate in new ways.

Many Thanks

- Stephen Simms, Nathan Heald, Justin Miller, Eric Isaacson, Matt Link, Robert Henchel, Scott Michael, Bret Hammond (IU)
- Kit Westneat (NYU)
- Eric Barton, Andreas Dilger, Robert Read (Whamcloud)
- The Technical Working Group of OpenSFS
- The Board of OpenSFS