INDIANA UNIVERSITY

Solution Architecture For UID/GID Mapping

Revision History

Date	Revision	Author
2012/08/01	1	jjw
2012/11/27	2 (formatted)	jjw
2012/12/05	3 (updated use cases and acceptance criteria)	jjw

Table of Contents

Introduction

Lustre is often used as a common work file system among several different compute clusters, allowing for different systems to have a fast, shared file space. Currently, to use Lustre across different compute clusters requires those compute clusters to share a common UID/GID space to maintain correct POSIX file system access control.

Within some organizations (such as universities), different departments maintain their own compute clusters but wish to use a centralized resource such as a common file system. To support this use case, Lustre must be able to properly map sets of UIDs to a canonical set based upon the NID of the mounting client. Indiana University developed such a system as a patched Metadata Server for Lustre 1.6 and Lustre 1.8.

The goal of this project is to extend Indiana University's solution in Lustre 2.X to include not only UID/GID Mapping in the Metadata Server, but the Object Storage Servers as well to support quotas with the canonical UID/GID space, as well as provide tools to manage the UID/GID space.

Solution Requirements

Toggle and Configuration

UID/GID Mapping must be available as an option. When toggled off, the behavior of the file system will be as it is currently.

UID/GID Mapping must provide configuration parameters to specify whether a particular NID is to have its UID/GID space mapped to the canonical set or the UID/GIDs in the request (and subsequent response) are to be trusted.

A configuration parameter must be provided to specify for a particular NID what UID/GID to map to a UID/GID contained in a request and subsequent response if it does not exist in the map.

A configuration parameter must be provided to specify whether a particular NID will allow (if its UID/GID space is trusted) UID 0 to perform operations on the file system. This will supersede the existing root squash mechanism.

ID Mapping

UID/GID Mapping must provide the ability for MDS and OSS nodes to translate between client UID/GIDs and canonical UID/GIDs to permit clients with a heterogeneous UID/GID space to mount the file system while maintaining and respecting POSIX file ownerships, permissions, and ACLs. Canonical UIDs and GIDs must be maintained on the OSS nodes to respect and maintain quotas.

Management Interface

UID/GID Mapping must provide a single master copy of the map on the Management Server, where each Metadata Server and Object Storage Server operates from a copy of the map that is kept in sync with the master. Administrative operations must take place only on the master copy of the map. The management interface must be able to perform the following operations while the file system is still mounted by clients.

- Add NIDs or groups of NIDs to the map
- Remove NIDs or groups of NIDs from the map
- Add a UID or GID to the map
- Remove a UID or GID from the map
- Force an update of the MDS and OSS with a new version of the map

Performance

The UID/GID Map must be structured to ensure that lookups to the map do not adversely affect the file system operations within which they are taking place.

Memory Usage

Both the canonical UID/GID Map and the working copies must be structured to minimize memory consumption

Use Cases

Inter-Organization Collaboration

Researchers from multiple sites are analyzing large images taken from a telescope using local compute resources. The compute resources are managed separately by each researcher's organization, and each site's policies differ in terms of account management. Copying the images back and forth would be time consuming and inefficient. Storing the images on a Lustre file system with UID/GID mapping and mounting the file system on resources local compute resources allows researcher to analyze the data in near real time, with file ownership and group permissions respected.

Inter-Department Collaboration

Researchers in the Chemistry and Computer Science departments of a large university have received funding for small cluster to perform interdisciplinary research. Both researchers want to make the data that they've collected available on resources local to the Chemistry and CS departments, which are managed by each department staff.

By mounting a centrally managed file system on each of the resources, and mapping the UIDs and GIDs to the appropriate values, allows each researcher to analyze and share data without exposing data owned by any other user or group.

Personal Workstation

A graduate student with a Linux workstation wants to use her advisor's data for visualization. Her hard drive would not be large enough to contain the data necessary for the visualization. Mounting the universities central Lustre file systems and UID/GID mapping her workstation allows her to easily perform the necessary work on the data set to produce visualization results.

Batch Job Processing

A compute resource has a Lustre file system mounted from another site across the wide area network. The workload on the compute resource is substantial and jobs need to be submitted to the queue several weeks in advance. A researcher can submit his or her job with data stored on the remote Lustre file system without needing a work-around to manage stored Kerberos credentials if the file system is mounted and UID/GUID mapping enabled and configured.

Solution Proposal

In current Lustre implementations, the UID/GID of the client request is either trusted or mapped to a per-NID linked list structure that is populated using timed Kerberos credentials. This creates problems in common use cases where jobs submitted against the file system do not run on machines with the same credentials, run after the credentials have expired or are limited to machines that share a homogeneous UID/GID space.

To avoid these problems, we are implementing a map that is configurable from the Management Server and statically maps NID:[UID/GID] to a canonical file system UID/GID. Additionally, user space tools, in the form of additions to lctl, will be written to control the configuration of the map and trigger the map update mechanism.

Map Structure

To both reduce map size and maintain lookup performance, the map will be keyed on both the NID of the client and the UID or GID of the incoming request or outgoing response. This will allow the map to be keyed for NIDs in contiguous ranges that can be abstractly called clusters, with common policies regarding UID and GID mapping.

Map Update

At boot, each Metadata Server and Object Storage Service will obtain a lock on the canonical UID/GID map residing on the Management Server. Obtaining the lock provides the MGS with a callback function that will be called at the time a map update is available.

The map will be structured on the MGS as a linked list of maps with serial numbers. Each update to the map will produce a new canonical map that is written to disk. Only the latest update to the map is kept on disk.

During the update, the Metadata and Object Storage Servers will provide the serial number of the map version that is currently resident. The Management Server will use this number to compute the delta between that version and the canonical master. Only the delta will be passed back to the MDS and OSS nodes.

Unit/Integration Test Plan

The following modes should be tested with both quotas disabled and enabled, and with each map operation (NID/UID/GID added and removed) between file system operations:

1. UID/GID Mapping toggled off
2. UID/GID Mapping toggled on with trusted cluster and admin disallowed
3. UID/GID Mapping toggled on with mapped cluster and admin disallowed
4. UID/GID Mapping toggled on with trusted cluster and admin allowed
5. UID/GID Mapping toggled on with mapped cluster and admin allowed
6. UID/GID Mapping toggled on with one trusted cluster, admin disallowed and one mapped cluster, admin disallowed
7. UID/GID Mapping toggled on with one trusted cluster, admin allowed and one mapped cluster, admin disallowed

For each mode, there will be four directories.

1. Directory owned by a UID/GID mapped only to cluster 1
2. Directory owned by a UID/GID mapped only to cluster 2
3. Directory owned by a UID/GID mapped to both cluster 1 and 2
4. Directory owned only by root

Permissions on each directory will be cycled through RWX for owner, group, and world permissions, as well as adding POSIX ACLs for an additional UID/GID mapped user for each directory. From each cluster, from a UID/GID mapped only on cluster 1, a UID/GID mapped only on cluster 2, a UID/GID mapped on both clusters 1 and 2, and a root user from both clusters 1 and 2, in each directory, the results of following operations will be tested for expected results for the mode. The results checked will be the success/failure of the operation, the permissions of any resulting files, the content of any resulting files, and any appropriate changes in quota.

1. File creates
2. File writes
3. File reads
4. File unlinks

Tests will also be made to updates to the UID/GID map and propagation of the map from the MGS to the MDS and OSS nodes. The following tests will be conducted from the MGS, and checks made on the MGS and OSS nodes to ensure that the map has propagated.

1. Add a cluster
2. Add an NID range to a cluster
3. Delete an NID range from a cluster
4. Change all configuration options of a cluster (root squash ID, trusted mode, admin mode)
5. Add a UID mapping to a cluster
6. Delete a UID mapping from a cluster
7. Add a GID mapping to a cluster
8. Delete a UID mapping from a cluster

All tests will be integrated into the test-framework.sh in the Lustre distribution.

Acceptance Criteria

UID/GID Mapping will be accepted as functioning properly if under each of the following conditions, file operations (create, read, write, unlink) are performed with file ownerships and permissions respected and enforced and (if applicable) quota operations performed correctly.

1. SFS-DEV-002.1.AC1:
   Mounting clients with UID/GID mapping toggled off. There is no change is current behavior.
2. SFS-DEV-002.1.AC2:
   Mounting clients with UID/GID mapping toggled on with the clients configured as trusted function. There is no change in current behavior.
3. SFS-DEV-002-1.AC3:
   Mounting clients with UID/GID mapping toggled on and mapping of UIDs and GIDs done for a subset of users on a subset of clients. Unknown users and groups are mapped to a squashed UID/GID, configured on a per cluster basis.
4. SFS-DEV-002-1.AC4:
   A subset of mounting clients configured for admin (root) access. Root access for all other clients are mapped to a squashed UID, configured on a per cluster basis.
5. SFS-DEV-002-1.AC5:
   All updates to the maps from the MGS can be made and reflected with the file system active.