The Flexible Files Layout

Benny Halevy <bhalevy@tonian.com>

IETF NFSv4 Working Group Meeting
Berlin, 2013-07-31

Status

- `draft-bhalevy-nfsv4-flex-files-00` submitted on July 15, 2013
- Rewrite of `draft-bhalevy-nfs-obj`
  - Based on WG feedback from IETF 85 meeting, Nov. 2012
  - And on Tonian implementation experience

Motivation

- **Flexible, per-file striping patterns, based on pnfs-obj (RFC5664) used for:**
  - Arbitrary data placement policies
  - E.g. Load balancing, life cycle management
- **Support for legacy NFS Data Servers**
  - Ubiquitous, standard protocol
  - Encourage best-of-breed solutions
  - Standalone, simple Data Servers (a-la Object Storage Devices)
- **Exporting of existing clustered File Systems**
  - For example: Ceph, GlusterFS
  - These file systems do not have a standard storage access protocol. Therefore NFS can be used instead

Changes since previous draft

- **Removed Support for the T10-OSD storage protocol**
- **Renamed**
  - **Capture spirit - propose an evolutionary step to the file layout rather than the object layout**
  - **Accommodate to WG naming convention**
- **Clarified state, security, and locking models**
  - For NFSv3, v4, and v4.1 (clustered) Data Servers
  - Documented security caveat for using NFSv3 with AUTH_SYS rpc auth.
- **Revised on-the-wire data structs**
  - NFS only Device Info and component creds
  - Device multi pathing compatible with NFSv4.1 files layout
  - Removed support for nested striping (simpler, server may use concatenated layout instead)
  - No need for layout-type specific LAYOUTCOMMIT layoutupdate4
- **Sparse/Dense striping patterns**
  - Compatible with NFSv4.1 files layout

Data Server Types

1) Standalone NFSv3 servers
2) Standalone NFSv4/4.x servers
3) Clustered NFSv4.x servers

State Model 1
Standalone NFSv3 Servers

- OPEN
  - No state on DS
- LOCK
  - Advisory only; via MDS
- Security and fencing
  - Arbitrary user/group file owner represent security token
    - Static 0640 mode; owner has RW access, group has RO.
  - Creds provided per component object with layout
  - Used to authorize client access and enforce security policy
  - Ownership changed by MDS on security attributes change or client fencing
  - Client MUST use ACCESS for each <open owner, layout>

State Model 2
Standalone NFSv4/4.x Servers

- OPEN / DELEGATION
  - MDS proxies OPEN / OPENCONFIRM to DS
  - passes DS stateid on layout to be used for I/O
  - Further OPENs may cause false fencing at DS and require refreshing LAYOUT
  - On OPEN_DOWNGRADE client needs to refresh the layout to get new DS stateid
- LOCK
  - Advisory only; via MDS
- Security and fencing
  - Real user/group file owner and mode used on component objects
  - Client uses real creds for I/O and these are authenticated by DS
  - Ownership/mode/ACL changed by MDS on security attributes change
  - Fencing achieved via CLOSE from MDS->DS (implicitly invalidates outstanding stateid)

State Model 3
Clustered NFSv4.x servers

- Compatible with the NFSv4.1 Files Layout
- Back-end control protocol used to implement global state
- DSs use I/O stateid to enforce OPEN/LOCK semantics
- LOCK
  - Mandatory locking possible via back-end protocol
- Fencing
  - Done via back-end protocol

To Do

- Editorial update queued (removed OSD leftovers)
- Need more discussion on WG mailing list
- Implement layout over standalone NFSv4 and clustered NFSv4.1 Data Servers

- Add to WG charter
  - What's left?

Future Work

- Standard back-end control protocol
  - We want NFSv4.x / pNFS to be ubiquitous; at least as NFSv3
  - We need to lower bar for smaller vendors
  - Expand ecosystem
- Client-based Copy On Write
  - a-la blocks layout (RFC5663), using separate read-only, read-write layouts with client-based on-demand zero fill and read-modify-write.

Resources

- http://tools.ietf.org/html/draft-bhalevy-nfsv4-flex-files
- http://tools.ietf.org/html/draft-ietf-nfsv4-rfc5664bis-01