NetApp™

Go further, faster

NFSv4.1 Linux Implementation Status

NFSv4 WG IETF 79 Beijing, China
Andy Adamson
andros@netapp.com

A Bit of History

- Linux has been a prototype platform for NFSv4.1 client and server since 2004
- Purpose is to give implementation experience back to the specification process
- Prototype followed the significant changes in draft-ietf-nfsv4-minorversion1versions (00 to 29)

A Bit of History

- As a prototype Linux implemented many features
  - Client and server sessions
  - pNFS client and server designed to accommodate file, object, and block layout services
  - pNFS device notification
  - Directory delegations
  - SSV

NFSv4.1 Client

- Minimal but complete sessions client in Linux 2.6.31
- No other optional features (pNFS slide later)
  - No Directory Delegations
  - No SSV
  - No deleg wants, fs_locations_info, etc
- Focus on integration with existing NFSv4.0
  - Especially state recovery
- Big hammer error recovery
  - Re-establish session on most errors

NFSv4.1 Client Sessions

- Fore channel
  - Single session per superblock
  - No trunking (clientid, session)
  - CB_RECALL_SLOT implemented
- Back channel
  - Shares fore channel connection
  - Single slot
  - No DRC, ca_maxresponsesize_cached set to 0
- Kerberos supported on fore channel
  - With AUTHS_SYS on back channel

NFSv4.1 Client Sessions

- Destroy and create a new session on:
  - Any back channel errors
  - Any session errors
  - Loss of connection
- State management
  - Re-establish client ID (and re-establish session) on most SEQUENCE status bit errors.

pNFS Client

- Designed as a generic piece which is part of the NFSv4.1 code base and ‘layout drivers’ for the file, object and block layout types each in their own kernel module.
  - Multiple concurrent layout modules supported
- Kernel submission starts with the file layout (simplest) and the generic pieces needed for its support.
- Object and block to follow

pNFS File Layout Client

- Whole file layouts only
  - No layout segments
- Forgetful client model
  - Avoid book keeping the races involving CB_LAYOUTRECALL, LAYOUTGET and LAYOUTRETURN
  - CB_RECALL_ANY will not return any layouts
- LAYOUTRETURN only on return-on-close and inode destruction (umount)

pNFS File Layout Client

- Large (multiple page) GETDEVICEINFO
- Device ID reaped when last layout reference disappears
- GETDEVICELIST not implemented
  - Wait until block layout driver
- CB_NOTIFY_DEVICEID not implemented

pNFS File Layout Client

- Code divided into ‘waves’ for submission
- First wave merged upstream into Linux 2.6.37
  - Loading of file layout driver
  - LAYOUTGET, GETDEVICEINFO
  - Layout cache, device ID cache, data server cache
- Four more waves planned before PNFS File Layout Client is complete.

NFSv4.1 Server

- Many features integrated into Linux 2.6.32
- Working on the TODO list to complete the mandatory feature set
- Server mandatory feature set larger than the client set
- Maintainer Bruce Fields requires a complete (all mandatory features) sessions implementation before reviewing any pNFS server code

NFSv4.1 Server Sessions

- Fore channel
  - Limit on total DRC memory footprint
    - Hand out sessions accordingly
  - Multiple sessions per client ID
  - Incomplete trunking
  - Kerberos support
- Back Channel
  - Only use CB_SEQUENCE and CB_RECALL
  - No Kerberos support

pNFS Server

- Designed as a generic piece which is part of the NFSv4.1 code base and an API for (per layout type) pNFS exportable file systems.
- Still in the prototype stage
- In-kernel pNFS exportable file systems include
  - GFS2 supports a file layout pNFS capable file system that hands out read iomode layouts only
  - Exofs supports an object layout pNFS capable file system

NFSv4.1 Server

- What to do when a feature is not supported
  - SSV and MACH_CRED currently return NFS4ERR_SERVERFAULT
  - Attempt to set unsupported backchannel security currently returns NFS4ERR_SERVERFAULT
  - Unsure what to do when client sets ACL retention bits

Lessons Learned

- It's taken some work on both the client and the server to figure out what the minimum feature set is to be spec compliant.
- How do we make it clear what the mandatory core of a new minorversion is?
- How do we keep that mandatory core small?
- How do we ensure negotiation of optional features is always clear?

Questions?

For more information

Client Sessions:

http://wiki.linux-nfs.org/wiki/index.php/Client_sessions_Implementation_Issues

PNFS File Layout Client:

http://wiki.linux-nfs.org/wiki/index.php/Client_pnfs_deliverables

Server Sessions:

http://wiki.linux-nfs.org/wiki/index.php/Server_4.0_and_4.1_issues

Server PNFS:

http://wiki.linux-nfs.org/wiki/index.php/Server_pNFS_issues