SNIA DEVELOPER CONFERENCE

SDC 24

BY Developers FOR Developers

September 16-18, 2024
Santa Clara, CA

The Linux NFS Server in 2024

Features and Plans

Chuck Lever, Oracle

Safe Harbor Disclaimer

- This presentation is intended to describe features that are part of an open source code base. It is intended for informational purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions.

- tl;dr: This talk is about upstream Linux.

Travelogue

- Introducing the Linux in-kernel NFS server
- Security features
- New facilities
- Upstream Continuous Integration

(Re) Introducing the Linux in-kernel NFS server

Current Feature Set

Basic Features

- NFS versions 2, 3, 4.0, 4.1, 4.2, and newer extensions
  - NFSv4.2 features: READ_PLUS, ALLOCATE, COPY offload, security labels
  - Extensions include Linux user extended attributes
- UDP, TCP, and RDMA transports
  - Supported RDMA fabrics include RoCE, InfiniBand, iWARP, and OPA
  - Software-emulated RoCE and iWARP on standard Ethernet devices
- Tier-1 filesystems: xfs, btrfs, tmpfs, ext4, and NFS re-export
  - Ceph, gfs2, and FUSE are also supported
  - Out-of-tree file systems like ZFS work but are not officially supported

Additional Capabilities

- NFS with Kerberos (authentication, integrity, privacy)
- NFSv4 referrals
- pNFS block / SCSI / NVMe (RFCs 5663, 8154, and 9561)
  - Simple/base devices only
- Observability via Linux kernel's ftrace facility
- Local POSIX ACLs are translated to NFSv4 ACLs on the wire

New Security Features

Kerberos, RPC-with-TLS, and more

RPCSEC GSS Kerberos

- Recently deprecated encryption types
  - Support for all DES and 3DES-based types has been removed
    - des-cbc-md4, des-cbc-md5, des-cbc-crc
    - des3-hmac-sha1
- Remaining encryption types
  - RFC 3962 (AES-1)
    - AES-128 with SHA-1
    - AES-256 with SHA-1

RPCSEC GSS Kerberos

- New encryption types
  - RFC 6803
    - Camellia-128 with CMAC
    - Camellia-256 with CMAC
  - RFC 8009 (AES-2 required for FIPS-140)
    - AES-128 with SHA-256
    - AES-256 with SHA-384
- Kunit tests added for all encryption types
- x86_64 acceleration support in plan
  - We plan to use AES-NI for the new ETM enctypes

RPC With TLS

- RFC 9289
  - TCP only for now; maybe DTLS eventually
  - Service levels
    - Encryption-only
    - Encryption with mutual peer authentication
  - Peer authentication
    - x.509 certifications (self-signed is supported)
    - PSK eventually
  - TLS Record Protocol is implemented via ktls
    - Linux in-kernel crypto can be used for broad hardware compatibility
    - NIC encryption offload can be used for good performance

Memory safety

- XDR decoding of received network data is critical
  - Old-school C macros have been replaced with xdr_stream utilities, which handle buffer bounds checking as part of decoding each data item
  - The future use of machine-generated code for XDR can help reduce possibility of programmer error
  - Re-implementing the RPC receive stack and XDR decoding in Rust might improve memory safety even further

Continuous Integration

kdevops, LTS kernels

Introducing kdevops

- kdevops: continuous integration for Linux kernel filesystems
  - Written and maintained by Luis Chamberlain <mcgrof@kernel.org>
  - Originally built for running the fstests suite for XFS and other local file systems
  - Uses Ansible and libvirt/terraform
    - Provisions test systems quickly and repeatably
    - Drives test runners
    - Accrues results

- https://github.com/linux-kdevops/kdevops

Improving the Quality of Upstream NFSD

- We've spent the past year developing NFS-specific workflows for kdevops
  - There are now five kdevops workflows that can set up distinct NFS clients and a local NFS server
  - Workflows include fstests, git regression, nfstest, ltp, and pynfs
  - Can provision NFS with Kerberos, TLS, pNFS, or RDMA
  - I test NFSD in the fs-next and fs-current kernels daily
- We continue to work towards making the test results public and archived

Improving the Quality of NFSD in LTS Kernels

- The kdevops workflows can also test LTS kernels
  - All five NFS workflows can run against kernels back to v5.4.y
  - I test NFSD in the queue-6.x and queue-5.x trees daily

New Facilities

NFSv4 delegation, pNFS, and more

NFSv4 Delegation

- NFSD has implemented read delegation for a long while
- It now also supports write delegation for files opened read/write
  - Soon, also for files opened write-only
  - Write delegation enables NFS clients to accelerate file locking
  - Note that POSIX rules about mtime updates still require client writeback, though there is standards work ongoing to help this situation
- NFSD now supports the CB_GETATTR callback
  - To avoid recalling a write delegation, server can service a GETATTR by querying the client holding the delegation for the file size and change attribute

NFSv4 Delegation

- NFSD has implemented read delegation for a long while
- It now also supports write delegation for files opened read/write
  - Soon, also for files opened write-only
  - Write delegation enables NFS clients to accelerate file locking
  - Note that POSIX rules about mtime updates still require client writeback, though there is standards work ongoing to help this situation
- We're collaborating with the FreeBSD NFS community to prototype directory delegation
  - This is a type of read delegation that enables more aggressive client-side caching of directory contents (ie, READDIR and LOOKUP results)

Parallel NFS

- NFSD implements the pNFS block layout type
  - NFSD acts as a metadata server; I/O is performed directly to other data servers
  - Simple/base device types only for now
  - NFS clients access these data servers directly via SCSI or iSCSI
  - Recently added is support for NVMe as well
- NFSD also has a simple implementation of the pNFS flexfile layout type
  - The DS and MDS are always the same, so there is no real benefit to using it

Courteous Lease Extension

- Clients keep their lease alive on a server by sending a lease-renewal operation every so often
- NFSD has traditionally purged a client's open/lock state once the client stops renewing
- However, network partitions between client and server do not mean that either the client or server has crashed
  - No reason to actively purge open/lock state quickly
- After client lease expiry, NFSD now hangs onto a client's open and lock state until another client attempts a conflicting open, or the client explicitly signals it is done with it

File Timestamp Resolution

- NFSv3 ctime and NFSv4 change_attr
  - NFS clients watch these attributes to know when their cache has become stale
  - On the server, incrementing these attributes and then persisting the updated value is expensive
  - Thus shortcuts are taken (like reduced-precision timestamps)

- It's possible that multiple individual file modifications can happen but be reflected by only one ctime/change_attr update
  - NFS clients can therefore miss cache-invalidating file modifications

File Timestamp Resolution

- Solution: Multi-grain time stamps
  - File systems use a high precision timestamp, but update and record a file's ctime/ change_attr *only if* they know the current value has been observed
  - A file system-wide floor timestamp is maintained to prevent concurrent modifications of multiple files from appearing to happen out of order
  - These are modifications to the VFS: all file systems (local and network) can take advantage of multi-grain

Next Steps

COPY offload, LOCALIO, NFSv4 POSIX ACLs, and more

A Taste of What's to Come

- New administrative features
  - Automated nfsd thread count with better NUMA awareness
  - An nfsdctl command along with an NFSD netlink protocol
  - An nfsref command for managing NFSv4 referrals
    - Referral to hostnames, IPv4 addresses, IPv6 addresses, and to alternate NFS server ports
    - Referral can control the client's choice of transport (TCP or RDMA)
    - Referrals to multiple targets

A Taste of What's to Come

- NFSv4.2 COPY offload
  - Synchronous copy and clone of files local to the server are available today
  - Asynchronous copy offload has some issues, but should be back soon
    - The CB_OFFLOAD callback is not reliable; OFFLOAD_STATUS is needed
    - Clients may start an unbounded number of asynchronous COPY operations
  - Server-to-server copy relies on async copy, so it is currently fully disabled
    - Even if it were enabled, Linux distributors haven't embraced it due to security concerns
    - Secure server-to-server copy (via Kerberos or TLS) is still somewhat distant

A Taste of What's to Come

- Attribute delegation
  - Problem statement
    - Traditionally, file timestamps are managed by the NFS server's local filesystem
    - To conform with POSIX, NFS clients must push all outstanding dirty file data to get a clean timestamp update, which adds unexpected latency to stat(2)
  - draft-ietf-nfsv4-delstid extends NFSv4.[12]:
    - Add a delegation mode (similar to write delegation) that enables NFS clients themselves to manage file timestamps
    - When the client returns this type of delegation, it sends updated file timestamps to the NFS server

A Taste of What's to Come

- Direct access to draft POSIX ACLs via NFSv4
  - Current situation
    - The NFSv4 currently supports the NFSv4 flavor of ACLs, but not POSIX
    - Linux local file systems implement only draft POSIX ACLs
    - Mapping between the two models is difficult
  - What if:
    - The NFSv4 protocol was extended to enable access to draft POSIX ACLs?
    - Linux NFSD implemented this extension?

A Taste of What's to Come

- LOCALIO
  - When a client and server can tell they are on the same physical host (eg, in containers):
    - Non I/O operations still use NFS via the network
    - NFS I/O operations (READ and WRITE) can be performed via local file I/O
    - The mechanism supports mounts of all NFS versions
    - Host kernel still needs to apply all NFSD access policies (ie, export settings) to local file I/O

Scheduled for Deprecation

- NFSv2
- UDP
- AES-1 Kerberos enctypes
- Maybe NFSv4 minor version 0?

Please take a moment to rate this session.

Your feedback is important to us.