NFSv4 Protocol Development

Alex McDonald
NetApp
Abstract

- The NFSv4 protocol undergoes a repeated lifecycle of definition and implementation. The presentation will be based on years of experience implementing server-side NFS solutions up to NFSv4.1, with specific examples from NetApp and others. We'll examine the lifecycle from a commercial implementation perspective; what goes into the selection of new features, the work with the IETF NFS standards body, the development process and how these features are delivered, and the impact these features have on end users.

- We'll also cover the work of Linux NFS developers and provide suggestions for file system developers based on these and vendor experiences; and finally, we'll discuss how implementation and end-user experience feeds back into the protocol definition, along with an overview of expected NFSv4.2 features.
Learning Objectives

- Understand the NFS protocol & its application to modern workloads
- How NFSv4.1 is being implemented by vendors and Linux developers
- The differences between NFSv3 and NFSv4.1, pNFS, FedFS
- An overview of proposed features in NFSv4.2
  - The IETF NFS Working Group
NFS; Ubiquitous & Everywhere

- NFS is ubiquitous and everywhere
- NFS doesn’t stand still
  - NFSv2 in 1983, through NFSv4.1 in 2010
  - NFSv4.2 to be agreed at IETF shortly
  - Faster pace for minor revisions
- NFSv3 very successful
  - Protocol adoption is over time, and there have been no big incentives to change
- See White Papers, Tutorials and webcasts for NFSv4.x; details at www.snia.org
Evolving Requirements

- Economic Trends
  - Cheap and fast computing clusters
  - Cheap and fast network (1GbE to 10GbE, 40GbE and 100GbE in the datacenter)
  - Cost effective & high performance storage based on Flash & SATA

- Performance
  - Exposes single threaded bottlenecks in applications
  - Increased demands of compute parallelism and consequent data parallelism

- Powerful compute systems
  - Analysis begets more data, at exponential rates
  - Competitive edge (ops/sec)

- Business requirement to reduce solution times
  - Beyond performance; NFS 4.1 brings increased scale & flexibility
  - Outside of the datacenter; requires good security & namespace uniformity
Performance, Management and Reliability

- Random I/O and Metadata intensive workloads
  - Memory and CPU are hot spots
  - Load balancing limited to pair of NFS heads; originally designed for HA
    - Not a limitation of the NFS 4.1 protocol
- Compute farms are growing larger in size
  - NFS head can handle a 1000+ NFS clients
  - NFS head hardware comparable to client CPU, I/O, Memory
  - NFS head requires more spindles to distribute the I/O
- Reliability and availability are challenging
  - Data striping limited to single head and disks
  - Non-disruptive upgrades affect dual-head configurations
  - Access and connectivity is typically limited to a pair of NFS server heads
NFS & The Virtualized Datacenter

Name Space

/ VM DB

HV1 HV2 Srv1 Srv2 Srv3

Cluster Datastore
Mount Server:/

Hypervisor Cluster Nodes

HV1 HV2

Srv1 Srv2 Srv3

pNFS MDS Server
NFSv4 Major Features; Security

- Strong security framework
- Access control lists (ACLs) for security and Windows® compatibility
- Mandatory security with Kerberos
  - Negotiated RPC security that depends on cryptography, RPCSEC_GSS
NFSv4 Major Features; Namespace

- Uniform and “infinite” namespace
  - Moving from user/home directories to datacenter & corporate use
  - Meets demands for “large scale” protocol
  - Unicode support for UTF-8 codepoints
- No automounter required
  - Simplifies administration
- Enhanced by FedFS
NFSv4.1 Major Features; Sessions

- NFSv3 server never knows if client got reply message
- NFSv4.1 introduces Sessions
  - Major protocol infrastructure change
  - Exactly Once Semantics (EOS)
  - Bounded size of reply cache
  - Unlimited parallelism
- A session maintains the server's state relative to the connections belonging to a client
NFSv4 Major Features; Stateful Clients

- NFSv4 gives client independence
  - Previous model had “dumb” stateless client
  - Server had the smarts
- Pushes work out to client through delegations & caching
- Why?
  - Compute nodes work best with local data
  - NFSv4 eliminates the need for local storage
  - Exposes more of the backend storage functionality
    - Client can help make server smarter by providing hints
NFSv4.1 Major Features; Layouts

- Layouts
  - Files, objects and block layouts
  - Provides flexibility for storage that underpins it
  - Location transparent
    - Striping and clustering

- Examples
  - Blocks, Object and Files layouts all available from various vendors
NFSv4.1 Major Features; pNFS

- NFSv4.1 (pNFS) can aggregate bandwidth
  - Modern approach; relieves issues associated with point-to-point connections

- pNFS Client
  - Client read/write a file
  - Server grants permission
  - File layout (stripe map) is given to the client
  - Client parallel R/W directly to data servers

- Removes IO Bottlenecks
  - No single storage node is a bottleneck
  - Improves large file performance

- Storage-access protocol
  - Files (NFSv4.1)
  - Block (iSCSI, FCP)
  - Object (OSD2)

- Control protocol
  - Not covered by spec; no generally agreed upon characteristic

- Improves Management
  - Data and clients are load balanced
  - Single Namespace

Data Servers

NFSv4.1 Client (s)

pNFS protocol

Storage-access protocol

Metadata Server

Control protocol

Data Servers
pNFS Operations

- **GETDEVICEINFO**
  - Client gets updated information on a data server in the storage cluster

- **GETDEVICELIST**
  - Clients requests the list of all data servers participating in the storage cluster

- **LAYOUTGET**
  - Obtains the data server map from the meta-data server

- **LAYOUTCOMMIT**
  - Servers commit the layout and update the meta-data maps

- **LAYOUTRETURN**
  - Returns the layout; Or the new layout, if the data is modified

- **CB_LAYOUT**
  - Server recalls the data layout from a client; if conflicts are detected
pNFS – NFSv4.1 files access

- Client mounts and opens a file on the server
- Servers grants the open and a file stripe map (layout) to the client
- The client can read/write in parallel directly to the NFSv4.1 data servers
Client mounts and opens a file on the server

Server grants the open and a block map (layout) to the client

Based on the layout obtained (read or write); the client can read/write in parallel directly to the SCSI targets
pNFS – Objects Access Model

- Client mounts and opens Object
- Server grants the open and an object stripe map and object capabilities (layout) to the client
- Based on the layout obtained (read or write); the client can read/write in parallel directly to the OSD targets
Federated File System

- Uniform namespace that has local and geographically global referral infrastructure
- Accessible to unmodified NFSv4 clients
- Addresses directories, referrals, nesting, and namespace relationships

- Client finds namespace via DNS lookup
- Sees junctions (directories) and follows them as NFSv4 referrals
FedFS is a set of open protocols that permit the construction of a scalable, cross-platform federated file system namespace accessible to unmodified NFSv4[.1] clients.

Key points:
- Unmodified clients
- Open: cross-platform, multi-vendor
- Federated: participants retain control of their systems
- Scalable: supports large namespaces with many clients and servers in different geographies
FedFS Protocols

Namespace Management
1. NSDB Management (LDAP)
2. Junction Management (ONC RPC)

Namespace Navigation
3. Namespace discovery (DNS)
4. Junction resolution (LDAP)

NFSv4.*(unchanged)

NFSv4.*Clients
- DNS Server
- Administrative Host
- NSDB Server

NFSv4.* Servers
Benefits of FedFS

- Simplified management
  - Eliminates complicated software such as the automounter
- Separates logical and physical data location
  - Allows data movement for cost/performance tiering, worker mobility, and application mobility
- Enhances:
  - Data Replication
    - Load balancing or high availability
  - Data Migration
    - Moving data closer to compute or decommissioning systems
  - Cloud Storage
    - Dynamic data center, enterprise clouds, or private internet clouds.
Federated Namespace Example

The illusion:

- The user and application software see a simple, hierarchical namespace.

The reality:

- Behind the scenes, simple management operations allow data mobility for high performance, high reliability, and high availability.
FedFS Namespace Example

The user requests /home/alice:

1. The client attempts to access /home/alice on server foo.
2. Server foo discovers that home is a namespace junction and determines its location using the FedFS NSDB service.
3. Server foo returns an NFSv4 referral to the client directing it to server bar’s /users.
4. The client accesses /users/alice on server bar.
Linux Client and NFSv4.1

- Upstream (Linus) Linux NFSv4.1 client support
  - Basic client in Kernel 2.6.32
  - pNFS support (files layout type) in Kernel 2.6.39
  - Support for the 'objects' and 'blocks' layouts was merged in Kernel 3.0 and 3.1 respectively
- Full read and write support for all three layout types in the upstream kernel,
Linux Client and NFSv4.1

- pNFS client support in distributions
  - Fedora 15 was first for pNFS files
  - Fedora 16, 17 current
- Red Hat Enterprise Linux version 6.2, 6.3
  - “Technical preview" support for NFSv4.1 and for the pNFS files layout type
- Other Open Source
  - Microsoft NFSv4.1 Windows client from CITI
  - Proposed for FreeBSD9.2
Client Support for Referrals

- For FedFS, requires NFSv4 clients supporting referrals; for example:
  - AIX: referrals and replication (including failover) supported since 5.3 (released August, 2004)
  - HPUX: referrals supported since HP-UX 11iv3 with ONCplus B.11.31.03 (released May, 2008)
  - Linux: referrals supported since 2.6.18 (released September, 2006)
  - Migration/replication support under development
  - OpenSolaris: referrals supported since build 131 (released January, 2010).
New Features in NFSv4.2

- Server-Side Copy (SSC)
  - Removes one leg of the copy
  - Destination reads directly from the source

- Application Data Blocks
  - Allows definition of the format of file
  - Examples: database or a VM image.
  - INITIALIZE blocks with a single compound operation
    - Initializing a 30G database takes a single over the wire operation instead of 30G of traffic.
New Features in NFSv4.2

- Space reservation
  - Ensure a file will have storage available
- Sparse file support
  - “Hole punching” and the reading of sparse files
- Labeled NFS (LNFS)
  - MAC checks on files
- IO_ADVISE
  - Client or application can inform the server caching requirements of the file
Files, blocks, objects can co-exist in the same storage network
- Can access the same file system; even the same file
- NFS flexible enough to support unlimited number of storage layout types
  - Three IETF standards, files, blocks, objects
  - Others evaluated experimentally
  - Support for cloud technologies
- FedFS extends NFS out from the data center
- NAS vs SAN; no-one cares any more
  - IETF process defines how you get to storage, not what your storage looks like
  - Underlying pNFS implementations will vary substantially between storage vendors
Conclusions

- NFS has more relevance today for commercial, HPC and other use cases than it ever did
  - Features for a virtualized data centers
- Developments driven by application requirements
- Adoption slow, but will continue to increase
  - NFSv4 support widely available
  - New NFSv4.1 with client & server support
- NFS defines how you get to storage, not what your storage looks like
## The Four Reasons for NFSv4.1

<table>
<thead>
<tr>
<th>Functional</th>
<th>Business Benefit</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Security</strong></td>
<td>Compliance, improved access, storage efficiency, WAN use</td>
</tr>
<tr>
<td>ACLs for authorization</td>
<td></td>
</tr>
<tr>
<td>Kerberos for authentication</td>
<td></td>
</tr>
<tr>
<td><strong>High availability</strong></td>
<td>High Availability, Operations simplicity, cost containment</td>
</tr>
<tr>
<td>Client and server lease management with fail over</td>
<td></td>
</tr>
<tr>
<td><strong>Single namespace</strong></td>
<td>Reduction in administration &amp; management</td>
</tr>
<tr>
<td>Pseudo directory system</td>
<td></td>
</tr>
<tr>
<td><strong>Performance</strong></td>
<td>Better network utilization for all NFS clients</td>
</tr>
<tr>
<td>Multiple read, write, delete operations per RPC call</td>
<td></td>
</tr>
<tr>
<td>Delegate locks, read and write procedures to clients</td>
<td>Leverage NFS client hardware for better I/O</td>
</tr>
<tr>
<td>Parallelised I/O</td>
<td></td>
</tr>
</tbody>
</table>
Summary

- pNFS is the first open standard for parallel I/O across the network
  - Ask vendors to include NFSv4.1 support for client/servers
- pNFS has wide industry support
  - Commercial implementations and open source
- FedFS extends NFS & a single namespace across the data center and beyond
- IETF NFS WG will continue to innovate based on new technologies and end-use requirements
Question & Answer