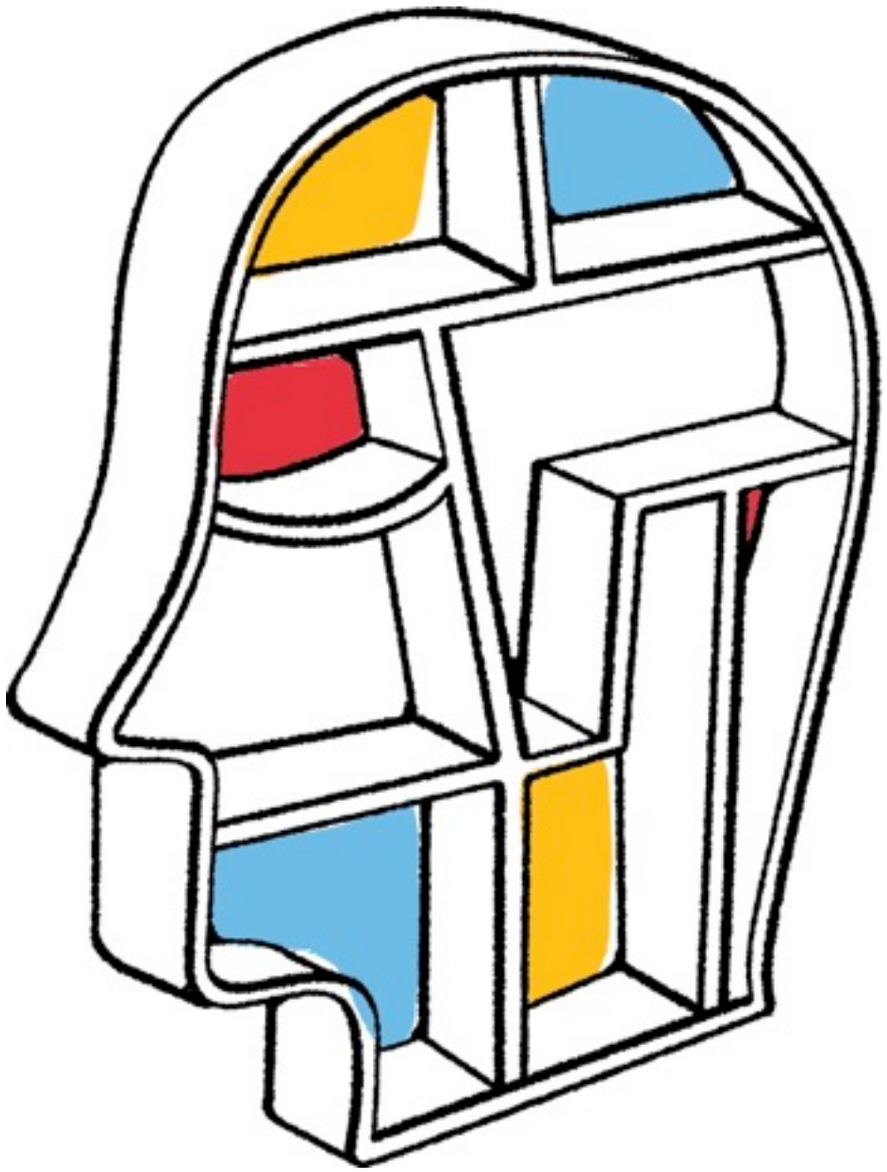
NetApp

Go further, faster®

NFSv4 Standards Update

Tom Haynes

thomas@netapp.com



Overview

- FedFS
- pNFS block disk protection
- 3530bis
- RPCSEC_GSSv3
- Labeled NFS Requirements
- NFSv4.2

FedFS

- Weave independently administered data servers into a namespace
- Documents in flight
  - Administration Protocol for Federated Filesystems
  - Using DNS SRV to Specify a Global File Name Space with NFS version 4
  - NSDB Protocol for Federated Filesystems
- Mostly done
  - Administrivia mode
  - No known blocking issues

pNFS block disk protection

- Very focused document on identifying block devices which belong to pNFS
  - I.e., a GUID inside the GPT which identifies the device as being used for pNFS
- Blazing a path through the process
  - Small
  - Focused

3530bis

- Has been in WG Last Call
- Several recent discussions on WG
  - Error codes on remove
  - client-persistent delegations
- Is a blocker for Charter Re-org

5664bis

- pNFS Object Layout
- Making the standard reflect implementations
  - Mostly refining RAID calculations
  - In queue behind 3530bis

RPCSEC_GSSv3

- Needed for
  - secure Server Side Copy
  - Labeled NFS subject labels
- Needs a driving force

Labeled NFS Requirements

- New draft is available
- Plenty of energy
- Needs reviewers

NFSv4.2

- Content is locked
- Delta from 4.1
  - Not a repeat of everything
    - Clarifications
    - New Content
  - Nothing is mandatory
  - 100 page goal versus 660
- Train model - goal is March 2012
- Needs an edit cycle to tie it altogether

Key differences from NFSv4.0 and NFSv4.1

- All new features are optional
- New NFSv4.2 server is simple
  - Feature not supported

- Provide features that will compel applications to adopt NFSv4.2

- Bringing local filesystem capabilities to the wire

- Non-posix semantics are creeping in

New direction for protocol changes

- “Instead of server vendors putting in new features that might attract application developers and vendors, they’re approaching server vendors requesting features that are available on local storage, but that you can’t get to currently via NFS”

What is in it?

- Clarifications
- Server Side Copy
- Sparse Files
- Seek Hole/Data
- Space Reservations
- Application Data Blocks
- Labeled NFS
- IO_ADVISE
- Change Attribute Behavior

Clarifications

- Address issues that cannot be done in a bis
- Intent is to be focused
- Minor XDR changes

Server Side Copy

- Traditional client based copy
  - client mounts source
  - client mounts destination
  - client reads a chunk from the source
  - client writes that to the destination
- A lot of overhead
  - Network: same data is transmitted twice
  - Client: Has to “switch” reply to a request
- Server side copy removes both of these

Protocol changes

- Defines the client-to-server interaction
  - I.e., setting up the transaction
- Does not define the server-to-server interaction
  - NFSv4.2 based copy
  - NDMP
  - scp
  - Proprietary transfer protocols

Security

- Pull based model

- The user on the client has privileges
  - Allowed to mount and read from the source
  - Allowed to mount and write to the destination

- Not some user, if any, on the destination
  - Destination may not be allowed to mount source
  - User account on destination might be different than that on the client

Security by intent

- Client preps source for a copy from a destination
  - COPY_NOTIFY
- Client informs destination to do the copy
- Two modes of authorization
  - RPCGSS_SECv3
    - Identity assertion
  - AUTH_SYS
    - Have to know the path to the open the file

Sparse files

- Exposes local filesystem's ability to have unallocated blocks to represent holes

- Allows a server to avoid sending allocated blocks which are all zeros

Seek hole/data

- A design effort has been to avoid space maps
- Solaris and Linux have system calls
  - seek_hole(offset)
  - seek_data(offset)
- Returns starting location of next hole/data

- We introduce SEEK
  - SEEK (stateid, offset, what)

Space Reservations

- Allow unallocated files to grow to set size
  - Avoid ENOSPC
- Allow detection of how much storage will be freed when a file is deleted
  - Dedup might have blocks shared
  - Hard to tell how much space will be reclaimed
- Dictate how large of a hole punch we will allow

Application data blocks (ADB)

- Allow applications to define virtual blocks in file
  - Databases
  - Virtual images
- Imposes a block header
- Rest of the block is zeros

INITIALIZE

- Sparse files
  - A newly created file has unallocated blocks
  - No need to INITIALIZE!
  - Not all servers support holes
  - Allows ZEROs to be written
    - Small RPC packet
    - Has to be asynchronous
- ADB
  - Send block header information
  - No need to store contents, unallocated block
    - Oops, has to be asynchronous

INITIALIZE (cont)

- Can be used to hole punch already allocated blocks
- Works with a discriminated arm of a switched union
- Servers are not required to support INITIALIZE
- Could support it for any combination of types

READ_PLUS

- Sparse files
  - minimal data transfer
- ADB
  - minimal data transfer

- Client might want to know if allocated or unallocated

- Data
  - Acts just like READ

Labeled NFS

- Normal UNIX security - DAC
  - Discretionary Access control
  - Mode bits and ACLs
- Want to be able to use MAC
  - Mandatory Access control
  - Finer grained
- NFS does not allow
  - passing a subject label
  - setting an object label
- Have to use another means
  - Database?

What is new for it in NFS?

- RPCSEC_GSSv3
  - Send a subject label
- NFSv4.2
  - Client can set an object label via an attribute
  - Server can inform clients of changes to object label via a callback
    - Only inform clients that have the file open

IO_ADVISE

- Hints for data access
  - *Server free to ignore*
- Derived from posix_fadvise()
  - Extensions based on non-posix experience
- IO_ADVISE operation (offset, length)
  - NORMAL
  - SEQUENTIAL
  - SEQUENTIAL_BACKWARDS
  - RANDOM
  - WILLNEED
  - WILLNEED_OPPORTUNISTIC
  - DONTNEED
  - NOREUSE
  - READ
  - WRITE

Change Attribute Behavior

- Client can determine server's method of updating the change attribute
  - Monotonic Increment
  - Version counter
  - Version counter -- No pNFS
  - Time Metadata
  - Undefined

Use cases or how does this drive adoption?

- Clarifications
  - Total geek factor
- Server Side Copy
  - Performance by avoidance of work
- Sparse Files
  - Virtual disk and database initialization
- Seek Hole/Data
  - Total geek factor? No.
  - Adopting common functionality in other OSes

Use cases (cont)

- Space Reservations
  - A hole is no good if it can't be filled
- Application Data Blocks
  - Allow applications to impose structure
- Labeled NFS
  - seVirt
    - Able to use with LUNs
- IO_ADVISE
  - Allow client to suggest server caching
- Change Attribute Behavior
  - A clarification

Resources

- What we are charted to do
  - http://datatracker.ietf.org/wg/nfsv4/charter/
- Where we document what we did
  - http://datatracker.ietf.org/wg/nfsv4/
- Archives of list discussions
  - https://www.ietf.org/mailman/listinfo/nfsv4
- The audience

Enjoy!

Standards Expert

What society thinks I do

What my mom thinks I do

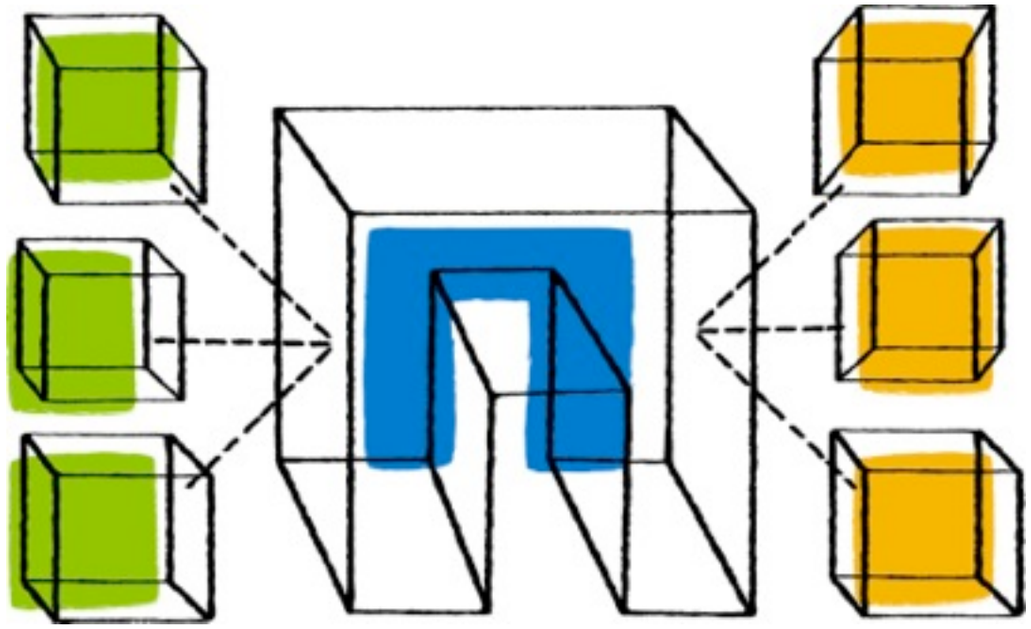
What my colleagues think I do

What my friends think I do

What I think I do

What I actually do

Backups



Protocol goals

- *Driven by user & application requirements*
  - Where do we get the requirements?
- Support sophisticated storage
- Minimalistic
  - Keep shared info between client & server to the minimum needed
  - Gives more flexibility to storage system