pNFS Directions / NFSv4 Agility

Matt Benjamin
Adam Emerson
Marcus Watts

Goal of the talk: stimulate discussion about

- Agility as a guiding vision for future protocol evolution
- NFSv4 protocol features that contribute most strongly to agility (and some that might be strengthened)

From the perspective of CohortFS, a startup and project attempting to build new functionality on (mostly) the LAYOUT concept from pNFS

NFSv4: A Standard File Access/Storage Protocol, that is *Agile*

- Incremental advances shouldn't require a new access protocol
- Capture more value from the engineering already done
  - Retain broad applicability
  - Yet adapt quickly to new challenges/opportunities

NFSv4 has delivered (over 10+ years of effort) on a set of features designers had long aspired to

- Significantly improved atomicity and consistency
- Vastly better integration of features (MOUNT, LOCK, etc)
- Referrals and single namespace (FedFS, NFSv4.2)

NFSv4 has sometimes been faulted for delivering slowly or imperfectly on some key promises

- Pervasive wire security that is both flexible and easy to use
- Capable and interoperable ACLs
- RDMA acceleration

NFSv4 has a set of interesting optional features not yet widely implemented, like

- Named attributes (we'll talk more about them)
- Read and write delegations (the former now becoming available)
- Directory delegations
- Secure State Verifier (SSV)
- Retention policy

Related Discussion in the NFSv4 Community (IETF)

The Minor Version/Extension debate

- de-serializing independent, potentially parallel extension efforts
- fixing defects in prior protocol revisions
- rationalizing past and future extension mechanisms

Related Discussion in the NFSv4 Community (IETF)

And critiquing old ones

- standardizing features on the blackboard, rather than in open prototypes
- leading to standard features we never implemented, or that do not work as expected

Lessons from the software engineering community

· Brooks – mythical man-month: adding more people to a project makes it later → people started documenting everything first, then coding

· Cathedral and the bazaar → documenting everything first takes forever and produces bad results; release early & often leads to better results

· But we knew this already! The original RFC process is much like the bazaar software dev model

Related Discussion in the NFSv4 Community (IETF)

The “Extensions” draft leaves many options open, but *prescribes*

- a process which supports development of new feature proposals in parallel (optional, mandatory, whatever)
  - capability negotiation
- experimental code point ranges for experimentation

Embracing Agility

- Noveck formulation (in “Extensions”) is subtle
  - rooted in NFS and WG history
  - future depends on the participants
  - can encompass but perhaps does not call out for an agile future

  (but we should)

CohortFS

Embracing Agility

- Capability negotation and experimental codepoint ranges strongly support agility
- Experimental ranges alone may not be sufficient
- What we really want is a model that encourages movement of features from
- <private experimentation> → <*shared experimentation*> → ? → <standarization>

CohortFS

Engineering Efforts Promoting Agility

- User-mode (and open source) NFSv4 Servers
  - Ganesha (v3, v4, v4.1!)
  - Others?
- And clients
  - CITI Windows NFSv4.1 client (hybrid)
  - Library client implementations (**)
    - Make available novel semantics and features (HPC, mobile devices, etc)

NFSv4 Protocol Concepts Promoting Agility

- We emphasize here
- NFSv4 has added concepts promoting extension flexibility in each revision since v3
  - Not just new RPCs and union types
- Opportunities to complete or widen concepts for greater flexibility/reach

CohortFS

COMPOUND

- combines operation grouping with context operations
  - grouping reduces op traffic w/equivalent semantics (see previous talk)
  - context evolves with operations, and inflects the operations
  - only a few file handle states and ops (PutFH/SaveFH/RestoreFH) defined
    - it could be pushed further (eg, stateids, even arbitrary properties)
      - reduced need for new ops

NFSv4 Protocol Concepts Promoting Agility

- Named Attributes
  - Support elaboration of conventions and even features above the protocol, with minimal effort and coordination
  - Have identity issues
    - Subfiles or proplists (**)
  - Namespace issues
    - System? User? Other?
    - Non-atomicity
    - Not inlined

LAYOUT

- powerful structuring concept carrying simplified transaction pattern
  - typed
- operations carry opaque data *nearly* everywhere
  - we should fix that (LAYOUTGET)
- application to data striping compelling
  - but has potential for broader application

Futures/Experimental Work

pNFS Striping Flexibility/Flexible Files (Halevy)

- New flexible files layout will add flexibility for per-file striping and specific parity applications to file layout
  - Pioneered in the OSDv2 layout
  - Presented at IETF 87

pNFS Metastripe (Eisler, further WG drafts)

- Scale-out metadata and parallel operation for NFSv4
  - Built on LAYOUT and attribute “hints”
  - CohortFS is prototyping metastripe on a parallel version of the Ceph file system

CohortFS

End-to-End Data Integrity (Lever/IBM)

- Add end-to-end data integrity primitives
  - Build on new READ_PLUS and WRITE ops (NFSv.4.2)
  - Optionally extend the perimeter of well-known error detection primitives to NFS
- Potentially high value for many applications
  - We certainly think so (below)

pNFS Placement Layouts (CohortFS)

- CohortFS is prototyping a design for algorithmic placement in pNFS layout extension
  - OSD selection and placement computed by a function returned at GETDEVICEINFO
  - client execution of placement codes, complex parity, volumes, etc

Replication Layouts (CohortFS)

- Client-based replication with integrity
  - eg, synchronous wide-area replication
  - Built on LAYOUT

Client Encryption (CohortFS)

- Relying (so far) on named attribute extension only
  - But could use named attribute atomicity
- Exploring optional OSD checksum operations for integrity checked operation over pNFS
- Hopefully, provided by end-to-end integrity already being worked on

Cache Consistency

- POSIX/non-CTO recently proposed (eg, Eshel/IBM)
  - Candidate for shared experimentation?
  - Potentially, more generality
    - eg, flexible client cache consistency models in NFSv4
      - add value to existing client caching implementations like CacheFS

New Participants

You?

References (Partial)

Noveck. NFSv4 Minor Versioning.
http://tools.ietf.org/agenda/87/slides/slides-87-nfsv4-4.pdf

Noveck. NFS Protocol Extension: Retrospect and Prospect.
http://tools.ietf.org/html/draft-dnoveck-nfs-extension-00

Eisler [et al]. pNFS Metastripe.
http://tools.ietf.org/html/draft-mbenjamin-nfsv4-pnfs-metastripe-01

Lever. End-to-End Data Integrity.
http://tools.ietf.org/html/draft-cel-nfsv4-end2end-data-protection-00

Halevy. pNFS Flexible Files.
http://tools.ietf.org/html/draft-bhalevy-nfsv4-flex-files-00

Eshel. Coherent Data Caching.
http://www.ietf.org/proceedings/87/slides/slides-87-nfsv4-2.pdf

Q/A