redhat

RGW-NFS:
An NFSv4 View of Ceph S3 Objects

Connectathon MMXVI

Intended Use Case:
Bridge legacy file-oriented application environments to S3

Presenter: Matt Benjamin
Title: Architect and Senior Manager
Date: March 3rd, 2016

Object Filesystems

Object Filesystems

Objects

- byte stream
- 0..n named attributes (addl. byte streams)

Just like objects in modern file systems

Semantics

NFSv4

- mutable
- byte-addressable (“READ <Offset, Length>”, “WRITE <Offset, Length>”, …)
- consistent (mostly)

Objects

- immutable
- atomic (“GET”, “PUT”, …)
- eventually consistent

Namespace (1)

NFSv4

/nfs4/foofs/

/foo/bar/baz

- /foo is a directory object (in the root of foofs)
- /foo/bar is a directory “object”
- /foo/bar/baz is a file (object)

Namespace (2)

S3

<bucket name> (“foo”) + <object>

(“bar/baz/quux/ranger/steve/....”,
“stuff”, “foo-1d0a66163426bdbfbc309cb65e4a7ccb-145692017 7”

- foo is a bucket
- bar/baz/quuz/ranger/steve/... is an object
- so is foo-1d0a66163426bdbfbc309cb65e4a7ccb-1456920177

Observations(1)

- Objects Implications (vs NFSv4)
  - Attributes are now important to apps (Hildebrand, 2016)
  - Namespace (key API stable point) often overhead
    - Mostly digested by NFSv4 community
  - Semantics
    - Unexamined assumptions, not stable (Opinion)
      - NFS vs POSIX
      - Eventual vs “Immediate”
      - Varying (“Workflows”, Hildebrand--but taking him out of context)

Observations(2)

- Objects v. File Dichotomy is not stable
  - >50% of distinction is API
    - POSIX v. Ad Hoc
    - Ad Hoc APIs are not stable

NFSv4 Pathless Objects (1)

IETF draft by Dipankar Roy [et al], NetApp (expired)

Relatively detailed proposal extending NFSv4 to support object access

- disjoint namespaces
  - objrootfh
  - lookup via searchattrs (lookup *is* search)
  - bucket -> filesystem
- exploratory semantics
  - presumption of object semantics, but lots of MAY

NFSv4 Pathless Objects (2)

Remarks

- no guidance on interaction of file and object namespaces (apparently disjoint)
  - no guidance on interaction of users with the NFSv4 protocol
    - possibly intended to be used in completely new operational style
      - APIs?
  - minimal guidance on semantics
    - suggests publishing objects with native semantics, e.g.,
      - atomic

What Users *Actually* Want (1) *(At Least for Now)*

Cross-Namespace Workflows

- attach Unix/workstation clients to object namespace
- propagate NFS objects to cloud
  - “get started”
  - streaming/push
    - e.g., of mostly-immutable data
  - archival
  - others

What Users *Actually* Want (2) *(At Least for Now)*

- bridge file semantics (in-place updates, random I/O)
  - “flash (or cache) and trash”
    - http://www.theregister.co.uk/2016/01/14/flash_and_trash_you_could_begin_with_cache_trash/

Other Object Unification Projects

SwiftOnFile

- expose (parts of) a distributed file system as objects

Remarks

- Internal system architecture detail, not intrinsically important whether eventually-consistent objects' origin is a distributed file system

But some interesting insights can be gleaned from the designers:
http://researcher.watson.ibm.com/researcher/files/us-dhildeb/hildebrand-fast16-tutorial.pdf

Ceph RADOS Gateway ("RGW")

Relatively rich interpretation of HTTP object storage for Ceph storage clusters

- Relatively uniform distribution of striped data across Ceph cluster (or portions thereof)
- Unified S3 and Swift namespaces
  - Objects and Buckets (Containers)
  - Consistent caching at RGW HTTP translators
- Flexible object layouts
  - Head -> Tail extent composition w/COW semantics, versioning (if desired)
- Atomic updates

See https://www.youtube.com/watch?v=zvfv3pXq0Ww

RGW-NFS

New NFS-Ganesha FSALexporting RGW Namespace. Runs a full Ceph RGW under an NFS-Ganesha process. Cache coherent peer of all gateways in the cluster.

Goals:

- Add RGW/S3 as an adjunct namespace on existing POSIX clients
- Support S3 Legacy “Prefix and Delimiter” Namespace Convention
  - Only supports ‘/’ delimiter currently
- Build foundation for more interesting workflows, e.g.
  - pNFS direct placement of sharded objects (future)
  - consistent caching (future)

Why Not S3FS?

"s3fs allows Linux and Mac OS X to mount an S3 bucket via FUSE"

https://github.com/s3fs-fuse/s3fs-fuse

Best effort answer:

- Namespace control
  - explore design space for S3 navigation (e.g., pathless)
- Cache Consistency
- Performance
  - knows RGW operation internals
  - exploit RGW object layouts (eventually)

Open Problems

Exposing S3 names in NFSv4 invites predictable friction--mostly arising from POSIX environments and clients:

- PATH_MAX
- Bucket -> Directory Mapping (Size)
- Synthetic Directories
- AuthN+AuthZ (Digests)

These are precisely the problems Pathless notation seeks to address. How might a POSIX client present pathless conventions?

Partial Solutions

- APIs
- Xattrs – done, let's make this a WG item
  - https://tools.ietf.org/html/draft-ietf-nfsv4-xattrs-01 (Naik, Eshel)
- more synthetic objects (like AFS .backup and Ceph .snaps)
- ideas?

redhat.

Questions?

Thank you