Fs_locations and all that Jazz

Migration and Replication in NFS-v4

Dave Noveck

dnoveck@netapp.com

March 4, 2002

NetworkAppliance®

Outline

Motivation for fs_locations attributes

Power of fs_locations concept

- Replication
- Migration
- Cross-server links (e.g. global name space)

File handle strategies for migration

Deployment issues and roadblocks

Possible extensions

NetworkAppliance®

Motivations for fs_locations

Some needed features

- Read-only replication (e.g. binaries)
- Migration
  - Load balancing
  - Continued access during server maint

Fs_locations specifies other locations of a fs

- What server
- Location (in pseudo-fs) within server

Features seemed to require something like it

- But it seemed so … un-NFS like

The power of fs_locations

Note that fs_locations gets top billing

V4 steps away from classical NFS ideals

- Statelessness
  - OPEN's and locking
- Mount containment
  - Pseudo-fs
  - per-owner+server sequencing
- Server containment (fs_locations)
  - Cross-server references
  - **Has the potential to be just as important as any of the others**

Replication

Fs_locations provides alternate locations

- Client fetches when crossing into fs

Upon failure

- Client looks for data at those locations
- Will generally find one that is up

Useful for binaries and other read-only data

Volatile filehandles suffice

Important issue:

- The replicas must actually be replicas
- No protocol support to assure this right now

Migration

A file system may move to another server

- Client gets NFS4ERR_MOVED error
- Fetches fs_locations attribute
  - only operation that doesn't get MOVED error
- Client then goes to specified server
  - Uses pathname to get to actual new fs

Then the fun begins

- Re-establishing state
- New filehandles for old?

Global name space via migration

How migration can deal with /afs-envy

Create a server to expose a name space

- Has been called a “referral server”
- e.g., Clients reference data starting at nfsv4:/
- Each fs in the name space can be found
  - But not referenced
- Each fs appears as migrated
  - fs_locations tells where it is

Clients don’t need their own location info

Referral servers can be replicated

Re-establishing state on migration

Two options:

- Server's moves stateid's
  - Client uses the old ones (and they just work)
  - Requires agreement between the servers
- Treated as server reboot
  - Client gets stale-stateid or stale-clientid error
  - Client uses reclaim logic
  - Still requires agreement between the servers
    - But much more limited

Peristent filehandles for migration

For homogeneous migration

- same vendor
- block-by-block fs copy
- or, move the disks (e.g.within a SAN)

It works

It has significant benefits

- Load balancing

But has limited applicability

Volatile filehandles for migration

Required for migration w/o image copy

Can be made to work

- But there are client costs
  - Keeping name information around

If you allow rename of open files (e.g. UNIX)

- You have to be careful
- Don’t want to open and write wrong file
  - It can ruin your whole day

Volatile filehandles, continued

WARNING: Spec is unclear on some of this

– Need to resolve and possibly fix

Maintain file handles for open files

- Expire all others
- Keep old file handles on on-disk table
- As files are closed, expire those handles
- On-disk table gets smaller
- And eventually is freed

Expire-type should be *volatile-at-any-time*

- With *no-expire-while-open* added

Deployment issues for migration

It isn't implemented yet

- Other things had higher priority
- But people are starting to work on it

For many uses, need ubiquitous V4 clients

- If you migrate, and have v3 clients
  - Need a solution for them (IP routing tricks)
  - But that solution will work for v4
- Until V4-only environments start to appear
  - Benefits of v4 transparent migration are fairly limited

Other roadblocks

Migration between server vendors

- No solutions yet
- Work has started on migration protocol
  - Just barely

Complexity of correct handle translation

- Manageable if carefully done
- Extensions (in a minor version) might help

Possible extensions (e.g. in v4.1)

Better filehandle management

- Avoid client overhead for volatile fh's
- Explicit filehandle trade operation?

Support fs split-up and merger

- Reduce disruption to clients

Other fs_locations-like attributes

- Allow single files to move (load-balancing)
  - file_locations attribute?
- Striping large files (scientific applications)
  - file_locations (array) plus stripe_size