Federated-fs protocol:

Overview and Implications for NFSv4

R. Tewari, M. Naik
IBM Almaden

D. Ellard, C. Everhart
NetApp

Outline

- Overview of Fedfs
- Fedfs and NFSv4
- Beyond namespaces
- Discussion

Background

- Idea: build a cross-platform federated file system with a shared common namespace.
  - Independent fileservers are federation members
  - Federation members are peers
  - Could have multiple administrative boundaries
- Use Cases:
  - Single enterprise with multi vendor fileservers and fileserver collections
  - Federated enterprises sharing independently managed namespace(s)
- Open Issues
  - Common namespace finding root fileservers not defined by protocol
  - Sophisticated Security Model
  - Access/authorization, ID mapping issues

Terms and Definitions

Terms and Definitions

- Fileset: the basic organizational unit for data
  - Container of data abstraction
  - To the user, it's a filesystem or a directory tree
  - Behind the scenes, can be implemented as a replicated, mobile storage container
- FSN: "Fileset Name"
  - Symbolic name of a fileset
  - UUID + Name of the namespace database (NSDB) responsible for the fileset
  - FSN is globally unique

Terms…

Terms…

- FSL: "Fileset Location"
  - Location of an implementation of a fileset
  - A fileset may be implemented by several FSLs
  - FSLs can come and go, change, etc.
- NSDB: "namespace database"
  - Repository of fileset and fileset relationship information
  - Keeps track of the mapping from FSNs to FSLs
  - Every FSN has a single authoritative NSDB
- Junction: binds a path within a fileset to a target FSN
  - Requires fileserver support
  - a junction can be viewed as a reference from a directory in one fileset to the root of the target fileset

How it all fits together

How it all fits together

- Filesets are the building blocks.

How it all fits together

- Filesets are the building blocks.
- Junctions define the relations between the filesets

How it all fits together

- Filesets are the building blocks.
- Junctions define the relations between the filesets
- Namespace built by fileset junctions

How it all fits together

- Filesets are the building blocks.
- Junctions define the relations between the filesets
- Namespace built by fileset junctions
- When an NFSv4 client accesses a junction

How it all fits together

- Filesets are the building blocks.
- Junctions define the relations between the filesets
- Namespace built by fileset junctions
- When an NFSv4 client accesses a junction
  - (which looks to the client like a directory):

How it all fits together

- Filesets are the building blocks.
- Junctions define the relations between the filesets
- Namespace built by fileset junctions
- When an NFSv4 client accesses a junction
  - (which looks to the client like a directory):
  - The fileserver finds the FSN of the junction's target fileset

How it all fits together

- Filesets are the building blocks.
- Junctions define the relations between the filesets
- Namespace built by fileset junctions
- When an NFSv4 client accesses a junction
  - (which looks to the client like a directory):
  - The fileserver finds the FSN of the junction's target fileset
  - The fileserver finds (from the FSN) the NSDB location responsible for that FSN

How it all fits together

- Filesets are the building blocks.
- Junctions define the relations between the filesets
- Namespace built by fileset junctions
- When an NFSv4 client accesses a junction
  - (which looks to the client like a directory):
  - The fileserver finds the FSN of the junction's target fileset
  - The fileserver finds (from the FSN) the NSDB location responsible for that FSN
  - The fileserver queries the NSDB for the current set of FSLs

How it all fits together

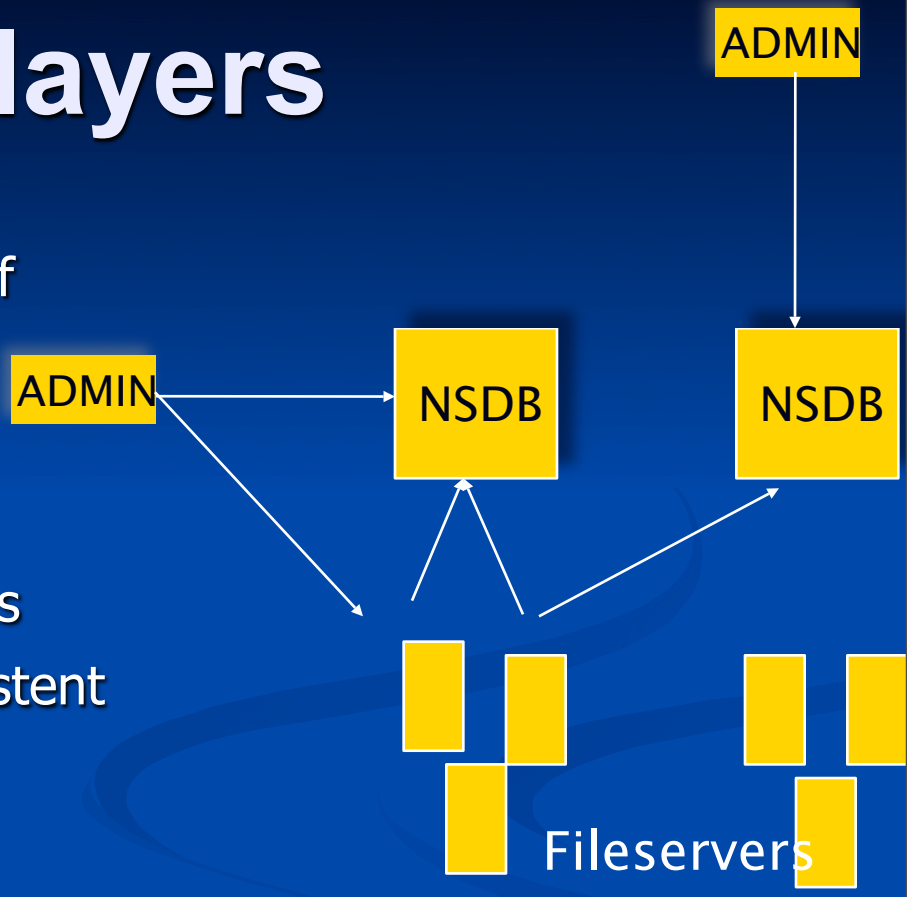
- Filesets are the building blocks.
- Junctions define the relations between the filesets
- Namespace built by fileset junctions
- When an NFSv4 client accesses a junction
  - (which looks to the client like a directory):
  - The fileserver finds the FSN of the junction's target fileset
  - The fileserver finds (from the FSN) the NSDB location responsible for that FSN
  - The fileserver queries the NSDB for the current set of FSLs
  - The fileserver uses the NFSv4 referral mechanism
    - informs the client that the directory has "moved"
    - Returns the FSL information in the fs_locations attribute

How it all fits together

- Filesets are the building blocks.
- Junctions define the relations between the filesets
- Namespace built by fileset junctions
- When an NFSv4 client accesses a junction
  - (which looks to the client like a directory):
  - The fileserver finds the FSN of the junction's target fileset
  - The fileserver finds (from the FSN) the NSDB location responsible for that FSN
  - The fileserver queries the NSDB for the current set of FSLs
  - The fileserver uses the NFSv4 referral mechanism
    - informs the client that the directory has "moved"
    - Returns the FSL information in the fs_locations attribute
  - The client is redirected to the root of the fileset that is the target of the junction

Three players

- Administrator
  - Initiates creation/updates/deletes of filesets, junctions, fileset locations
- NSDB
  - Maintains the fileset state
  - Responds to inserts/updates/queries
  - An NSDB is an LDAP server + persistent database
- NFSv4 server
  - In response to client requests, queries the NSDB for FSLs
  - In response to admin requests, updates/ queries server state (junctions)



Three protocols

- Admin → NSDB
  - Update the NSDB state
  - Implemented as LDAP using fed-fs schema
- Fileserver → NSDB
  - Query the NSDB
  - Implemented as LDAP queries using fed-fs schema
  - Why LDAP?
    - LDAP client easily available/integrated with NFS server
    - Well understood (although painful) protocol
    - Open source LDAP server implementations available, security supported, replication and recovery supported
- Admin → Fileserver
  - Update/query the NFSv4 server state (junctions)
  - Implemented via SUNRPC
  - Why not LDAP?
    - We don't want every NFS server to be an LDAP server as well (it can be

Fed-fs and NFSv4

Current V4 Mechanisms

- return NFS4ERR_MOVED on client access
- getattr (fs_locations, fs_locations_info, fs_status, change_info, fsid, mounted_on_fileid)
- Pure referrals
  - absent filesystem
- Replicas with multiple fs_locations
  - present filesystem with multiple locations for failover etc.
- Migration of existing fileset
  - or a newly carved out fileset
  - present filesystem becoming absent

NFSv4 server interaction

- Fileserver has knowledge of junctions but not of filesets or FSLs
  - It needs to recognize a junction
  - It needs have a mechanism to create a junction
  - It needs to store the FSN of the target fileset
- On a junction traversal
  - Find the NSDB from the FSN of the fileset referred to in the junction
    - `FSN= NSDBNAME:junctionKey`
  - Fileserver needs to make an LDAP query request to the server NSDBNAME
    - `LDAP Query: ((&objectClass=FslObject) (junctionKey=JUNCTIONKEY))`
  - The LDAP server returns fs_locations details in the fed-fs schema
    - `entries: dn: junctionKey=JUNCTIONKEY,nsdbName=NSDBNAME,ou=fed-fs`
      `fslHost: HOST`
      `fslPath: PATH …`

V4 server-related issues

- Should the server cache the fs_locations or query each time?
- Query based on junctionKey (it is a uuid) not pathname
- fs_locations contain server_name:/path not IP addresses
- Support LDAP client and parsing fed-fs schema

V4 Client Support

- v4 client issues
  - support for fs_locations with hostname and path
  - support for pure referrals
  - replica failover and migration across vendors
  - adding junctions to existing exported paths that the client has traversed
  - Caching of the FSL list?
- What is the state of the clients
  - Linux (pure referrals supported), AIX (pure referrals and replica failover), Solaris (pure referrals kind of)

Beyond Namespaces

Replication support

Replication support

- Should it be part of fed-fs protocol?

Replication support

- Should it be part of fed-fs protocol?
- Fed-fs, FSLs and replicas
  - FSLs map to the multiple replicas of a fileset
  - FSL consistency is not guaranteed by fed-fs
    - FSLs could be read-only or writable.
  - Assume an external replication protocol will update the list of "valid" FSLs and their versions in the NSDB?
  - NSDB will return to server a list of the current FSLs with attributes
- Failover
  - If a mount fails for the "first" FSL in the list proceed to next
  - If an FSL fails after the client has mounted the client needs to failover to another FSL in the list
    - Filehandles and fileids will change
    - Need support for volatile filehandles

Migration support

- Fileset migration
  - Fed-fs only creates/removes a junction reflecting a fileset migration
  - Migration of an existing FSL
    - fs_root is the root dir of the FSL
  - Migration of subdir within an FSL created as a nested fileset
    - fs_root is a subdirectory inside the FSL
    - Need to also reflect a change in fsid (filesystem split)
  - Need volatile filehandles (volume migration) for cross vendor support
  - For locks client handles it as a lock revocation/server failure
  - V4 client needs to handle new filehandles and fileids
- Migration seen as a useful tool in a federated system

Finding Root fileservers

- For a common namespace across the federation
  - Client needs to mount the root of the namespace from one of the root fileservers
  - From there on it could follow junctions and traverse the namespace
- DNS SRV records to find the root?
- Other techniques that do not rely on DNS?

Discussion questions

- Choice of protocol for fileserver—admin interaction
- FSL consistency not defined
- Caching of namespace information across NSDBs
  - Default local NSDB not defined
- Security Issues
- Access/authorization, ID mapping issues

Prototype Efforts

- NDAF on AIX in 2006
- Glamor on Linux in 2007
- Netapp

More Information at

- list federated-fs@sdsc.edu
- IETF Draft
  - draft-ellard-nfsv4-federated-fs-01.txt
  - draft-ellard-nfsv4-federated-fs-admin-01.txt
  - draft-tewari-nfsv4-federated-fs-protocol-01.txt
- Emails:
  - daniel.ellard at netapp.com
  - tewarir at us.ibm.com