NetApp™

draft-dipankar-nfsv4-pathless-objects-01

Go further, faster

Pathless Objects and Search Attributes

- Dipankar Roy

Key Issues Solved

- Object based storage with NFS.
  - Uses NFS filehandle as Object identifier
  - Filenames and pathnames are not required
  - Container based namespace
- Tag based searching for Objects or Files
  - Metadata for an Object or a File can be the searchable tags
  - Search and look up multiple objects with a single query and rich query semantics
  - Search for an object at the server instead of searching at the client

Additions to NFSv4

- Two new file types - NF4NOPATHOBJ and NF4OBJSET
- Two new operations – PUTOBJROOTFH and PUTSRCHATTR
- Two new attributes – sattrsupport and srchattrlist
- One new search query structure – srchquerylist
- Minor modifications to NFSv4 operations and structures that deal with pathnames

Considerations and Use cases

- Advisory locking must be supported, mandatory locking optional.
- Tag based filesystems are currently being used by major search engines, social networking websites, online sellers, multimedia websites etc.
- Several open source implementations available for tag based filesystems
- No overlapping content with any other RFC, as far as we know
- Prototyping is in progress
- Potential impact can be such that NFS may become protocol of choice for Object based storage

Action Items

- Who should be involved
  - Anyone with active interest in NFS
  - Preference to active members of NFSv4 charter of IETF and NFS client/server implementers
- Intended for NFSv4.2
- Review requested from WG
- Authors to work on review comments and prototype
- Targeted completion of review and prototype before next IETF

Questions/Answers

- Thanks to Thomas Haynes and Manjunath Shankararao for reviewing this presentation.
- URL: http://tools.ietf.org/html/draft-dipankar-nfsv4-pathless-objects-01
- Any questions can be sent to
  - nfsv4@ietf.org
  - dipankar@netapp.com
- Thank you!

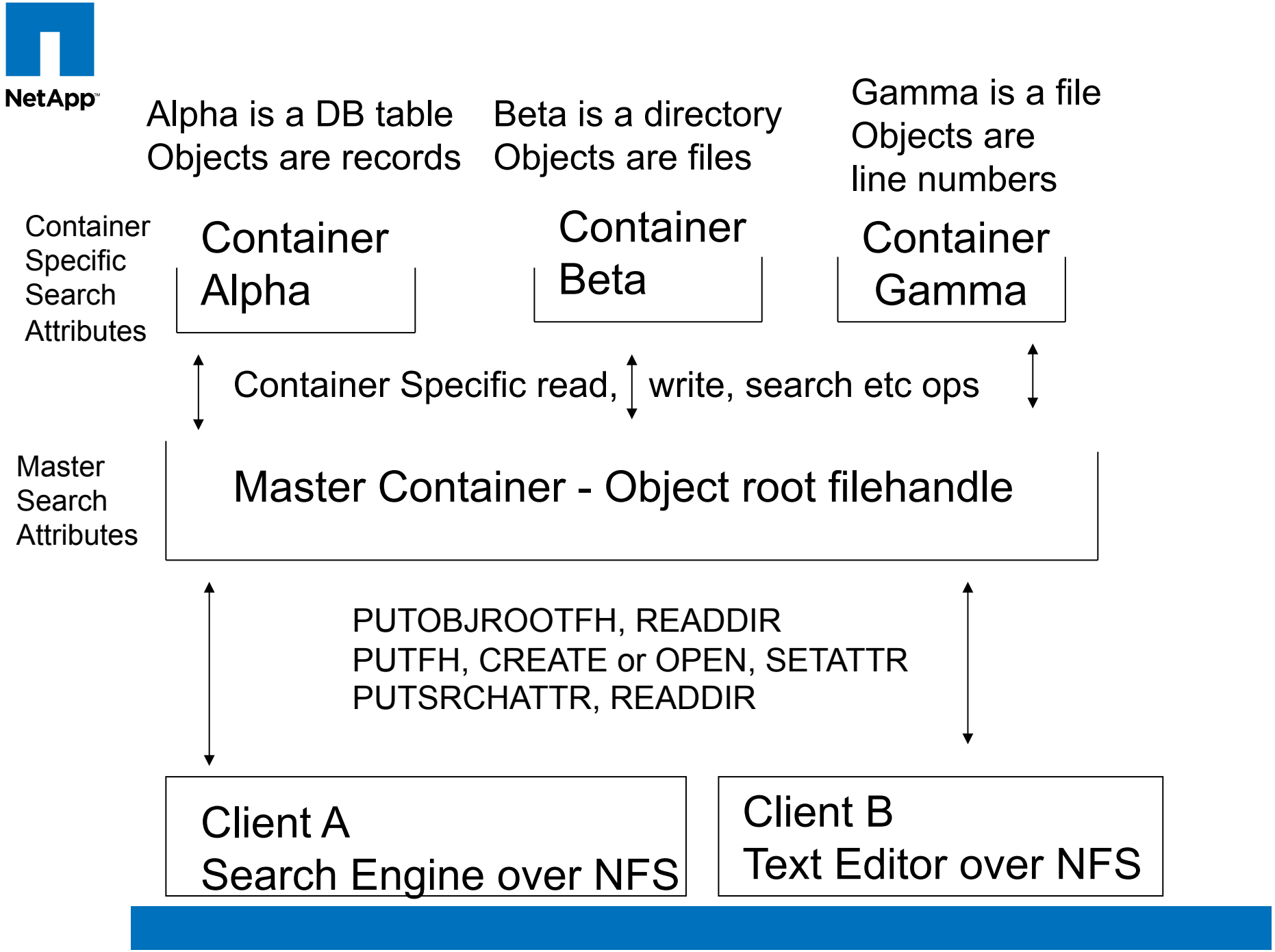
NetApp™

Go further, faster

Backup Slides

Pathless Objects and Object Sets

- Object Root Filehandle
  - Similar to NFS public root filehandle
  - Master container for Object Sets
  - Gives a new namespace for pathless objects
  - READDIR at Object Root Filehandle lists all Object Sets
- New file types
  - NF4NOPATHOBJ : For pathless objects
  - NF4OBJSET : For Object Sets
- Use existing NFS operations for creation and maintenance
- Object Sets have unique names but Objects do not
- Optional to support: file names, POSIX semantics, stateful operations, device files etc


Alpha is a DB table
Objects are records
Beta is a directory
Objects are files
Gamma is a file
Objects are
line numbers
Container
Specific
Search
Attributes
Container
Alpha
Container
Beta
Container
Gamma
Container Specific read, write, search etc ops
Master
Search
Attributes
Master Container - Object root filehandle
PUTOBJROOTFH, READDIR
PUTFH, CREATE or OPEN, SETATTR
PUTSRCHATTR, READDIR
Client A
Search Engine over NFS
Client B
Text Editor over NFS

Search Attributes

- Used to lookup pathless objects. Can also be used for regular files.
- New recommended attributes
  - sattrsupport : server supports search attributes
  - srchattrlist : the search attribute
- A search attribute is defined by the tuple

  <name, type , values>
- Type can be string or integer
- Used with SETATTR and GETATTR
- New operation to lookup objects based on search attributes - PUTSRCHATTR

Search Attributes XDR

- bool sattrsupport; /* indicates search attributes are supported */
- enum svaltype {
  SVAL_TYPE_NUM = 0; /* Search Attribute value is a number */
  SVAL_TYPE_STR = 1; /* Search Attribute value is a string */
  };
- union sval switch (svaltype type) { /* single search attribute value */
  case SVAL_TYPE_NUM: int64_t svalnum;
  case SVAL_TYPE_STR: component4 svalstr;
  default: void;
  };
- typedef struct sval svalist<>; /* array of attribute values */
- struct srchattr {
  component4 srchattrname; /* name of the search attribute */
  svaltype type; /* type of the search attribute */
  svalist srchvalist; /* list of values for this attr */
  };
- typedef struct srchattr srchattrlist<>;

Search Attributes Query

- Collection of search attributes matching one or more values
- Match can be based on equals, less than or greater than
- Queries are joined together using logical AND, OR and NOT operations
- Provision for embedded queries and ordered evaluation using priority
- Used in PUTSRCHATTR

Search Attributes Query XDR

- enum srelation {
  SRELN_EQUALS = 0;
  SRELN_GREATER = 1;
  SRELN_LESSER = 2;
  };
- enum srchqueryjointype {
  SQUERY_NONE = 0;
  SQUERY_AND = 1;
  SQUERY_OR = 2;
  };
- struct srchquery {
  srchattrlist search_attrs;
  srelation search_relation;
  srchqueryjoinype sqjtypenext;
  uint32_t priority;
  uint32_t flag;
  };
- typedef struct srchquery srchquerylist<>;

New Operations

- PUTOBJROOTFH
  - Similar to PUTROOTFH but for pathless Objects
  - READDIR following PUTOBJROOTFH lists all Object Sets
- PUTSRCHATTR
  - Current file handle must be the Object Root filehandle or the filehandle for an Object Set
  - Matches all objects specified in the search attribute query
  - Must be followed by a obligatory READDIR, which is used to structure the reply

Modifications to NFS operations

- CREATE
  - Must be used to create Objects Sets and may be used to create pathless objects
  - Unique name must for Object Set
  - Empty string is a valid name for a pathless object
- LOOKUP
  - If multiple files matching a name is found, LOOKUP returns an error
- OPEN
  - Can be used to create pathless objects with an empty name
- READDIR
  - Must be used immediately after a PUTSRCHATTR, returns all objects matching the query
  - Can return empty file names

Migration and Replication

- Supported with trivial modifications to fs_locations and fs_locations_info
- "rootpath" and "fs-root" in fs_location4 needs to be Object Set names.
- "fli_rootpath" and "fli_fs_root" for fs_locations_info4 contains Object Set names.