File System Extended Attributes in NFSv4

Manoj Naik
Marc Eshel

Connectathon 2015
February 25, 2015

Why do we need xattrs in NFS?

- Widely supported by most OSes/filesystems...
  - … but no standard specification
- Copying files with xattrs over NFS results in loss
- Strong interest in community
  - Different semantics from named attributes
  - Needs well-defined specification for requirements and interoperability

What has transpired so far...

- 10/2013: Mailing list discussion on xattrs
- 11/2013 IETF 88: Should we add xattrs in NFSv4?
  - Yes, widely used and well supported
  - Since xattrs cannot be easily mapped to existing attributes in NFS, data loss occurs today if file with xattrs is copied over NFS
- 3/2014 IETF 89: First draft
  - Extend existing bitmap4, propose other options
  - Consensus to define new operations for xattrs
- 7/2014 IETF 90: Second draft
  - Propose new operations (GETXATTR, SETXATTR)
- 10/2014: More mailing list discussion
  - Feedback on use cases, operations, no consensus

What do we propose?

- Protocol enhancements to support xattrs
  - Clear interfaces for get/set/list/remove
  - Well-defined semantics
- Only user-specified xattrs
  - Opaque to NFS clients and servers
  - Discourage non-interoperable implementations

Basic Operations

- Given a file, return a list of all of the file's assigned extended attribute keys (`listxattr`)
- Given a file and a key, return the corresponding value (`getxattr`)
- Given a file, a key, and a value, assign that value to the key (`setxattr`)
- Given a file and a key, remove that extended attribute from the file (`removexattr`)

Protocol Enhancements

- New RECOMMENDED attributes
  - Query xattr support
- New OPTIONAL operations
  - Get, set, list, remove xattrs
- Extensions to ACE Access Mask Attributes
  - New bitmask constants for the access mask field

New Attributes

- Extend bitmap4 for use with GETATTR

Name	Id	Data Type	Acc
maxxattrsize	82	uint32_t	R
xattrsize	83	uint32_t	R

  - `maxxattrsize`
    - Max size supported by file system
    - 0 if not supported
  - `xattrsize`
    - Total size of all xattrs for a given file
- No limits on number or size of individual xattrs

New Definitions

```
typedef utf8str_cis  xattrname4;
typedef opaque       xattrvalue4<>;


struct xattr4 {
   xattrname4     xa_name;
   xattrvalue4    xa_value;
};

const ACE4_GET_XATTRS  = 0x00200000;
const ACE4_SET_XATTRS  = 0x00400000;
```

GETXATTR

ARGUMENTS

```
struct GETXATTR4args {
    /* CURRENT_FH: file */
    xattrname4      name;
};
```

RESULTS

```
struct GETXATTR4resok {
    xattrvalue4     value;
};

union GETXATTR4res switch (nfsstat4 status) {
 case NFS4_OK:
     GETXATTR4resok    resok4;
 default:
     void;
};
```

SETXATTR

ARGUMENTS

```
enum setxattr_type4 {
        SETXATTR4_CREATE        = 0,
        SETXATTR4_REPLACE       = 1
};

struct SETXATTR4args {
        /* CURRENT_FH: file */
        setxattr_type4   type;
        xattr4           xattrs<>;
};
```

RESULTS

```
union SETXATTR4res switch (nfsstat4 status) {
 case NFS4_OK:
        void;
 default:
        nfsstat4 res4<>;
};
```

REMOVEXATTR

ARGUMENTS

```
struct REMOVEXATTR4args {
        /* CURRENT_FH: file */
        xattrname4   names<>;
};
```

RESULTS

```
union REMOVEXATTR4res switch (nfsstat4 status) {
    case NFS4_OK:
         void;
    default:
         nfsstat4    res4<>;
};
```

LISTXATTR: ARGUMENTS

```
ARGUMENTS
struct READDIR4args {
        /* CURRENT_FH: file */
        nfs_cookie4      cookie;
        verifier4        cookieverf;
        count4           xattrcount;
        count4           maxcount;
};
```

LISTXATTR: RESULTS

```
struct entry4 {
        nfs_cookie4     cookie;
        xattrname4      name;
        xattrvalue4     value;
        entry4          *nextentry;
};

struct xattrlist4 {
        entry4          *entries;
        bool            eof;
};

struct LISTXATTR4resok {
        verifier4       cookieverf;
        xattrlist4      reply;
};

union LISTXATTR4res switch (nfsstat4 status) {
 case NFS4_OK:
         LISTXATTR4resok  resok4;
 default:
         void;
};
```

Caching and Delegations

- Caching behavior similar to other attributes (not data)
- SETXATTR also modifies “change” attribute
- Clients without delegations
  - can cache (unmodified) xattrs, validate using change attribute
  - must write-through changes (synchronously), may need to wait for delegation to be recalled
- Owner of read/write delegations
  - can cache (modified) xattrs
  - respond with new “change” value to CB_GETATTR

What's Next?

- Document a use case that supports the need for xattr support in NFS
  - Solicit community help
  - Needs to be interoperable across vendors
- Generate consensus in the WG
  - Requirements, operations
- Reference implementation

Questions?

http://tools.ietf.org/html/draft-naik-nfsv4-xattrs-01