NFSv4 T. Haynes
Internet-Draft Hammerspace
Intended status: Standards Track April 01, 2018
Expires: October 3, 2018

Extending the Opening of Files in NFSv4.2
draft-haynes-nfsv4-delstid-00.txt

Abstract

The Network File System v4 (NFSv4) allows a client to both open a file and be granted a delegation of that file. This grants the client the right to cache metadata on the file locally. This document presents serveral refinements to both the opeing and delegating of the file to the client.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on October 3, 2018.

Copyright Notice

Table of Contents

1. Introduction

In the Network File System version4 (NFSv4) a client may be granted a delegation for a file. This allows the client to act as the authority of the file's metadata and data. In this document, we introduce some new semnatics to both the open and the delegation process which allows the client to:

- determine the extension of OPEN (see Section 18.16 of [RFC5661]) flags.
- during the OPEN procedure, get either the open or delegation stateids, but not both.
- detect an offline file, which may be located off premise.
- cache both the access and modify times, reducing the number of times the client needs to go to the server to get that information.

Using the process detailed in [RFC8178], the revisions in this document become an extension of NFSv4.2 [RFC7862]. They are built on top of the external data representation (XDR) [RFC4506] generated from [RFC7863].

1.1. Definitions

delegation: A file delegation, which is a recallable lock that assures the holder that inconsistent opens and file changes cannot occur so long as the delegation is held.

stateid: A stateid is a 128-bit quantity returned by a server that uniquely defines state held by the server for the client. (See Section 8 of [RFC5661])

1.2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Determining the Arguments to OPEN

<CODE BEGINS>

```
///
/// struct open_arguments4 {
///   bitmap4  oa_share_access;
///   bitmap4  oa_share_deny;
///   bitmap4  oa_share_access_want;
///   bitmap4  oa_open_claim;
///   bitmap4  oa_create_mode;
/// };
///

///
/// enum open_args_share_access4 = {
///    OPEN_ARGS_SHARE_ACCESS_READ  = 0;
///    OPEN_ARGS_SHARE_ACCESS_WRITE = 1;
///    OPEN_ARGS_SHARE_ACCESS_BOTH  = 2;
/// };
///

///
/// enum open_args_share_deny4 = {
///    OPEN_ARGS_SHARE_DENY_NONE  = 0;
///    OPEN_ARGS_SHARE_DENY_READ  = 1;
///    OPEN_ARGS_SHARE_DENY_WRITE = 2;
/// };
///
```

```
   ///
   /// enum open_args_share_access4 = {
   ///    OPEN_ARGS_SHARE_ACCESS_WANT_ANY_DELEG          = 0;
   ///    OPEN_ARGS_SHARE_ACCESS_WANT_NO_DELEG           = 1;
   ///    OPEN_ARGS_SHARE_ACCESS_WANT_CANCEL             = 2;
   ///    OPEN_ARGS_SHARE_ACCESS_WANT_SIGNAL_DELEG_WHEN_RESRC_AVAIL
   ///                                                   = 3;
   ///    OPEN_ARGS_SHARE_ACCESS_WANT_PUSH_DELEG_WHEN_UNCONTENDED
   ///                                                   = 4;
   ///    OPEN_ARGS_SHARE_ACCESS_WANT_DELEG_TIMESTAMPS    = 5;
   ///    OPEN_ARGS_SHARE_ACCESS_WANT_OPEN_XOR_DELEGATION = 6;
   /// };
   ///

   ///
   /// enum open_args_share_access4 = {
   ///    OPEN_ARGS_CLAIM_NULL          = 0;
   ///    OPEN_ARGS_CLAIM_PREVIOUS      = 1;
   ///    OPEN_ARGS_CLAIM_DELEGATE_CUR  = 2;
   ///    OPEN_ARGS_CLAIM_DELEGATE_PREV = 3;
   ///    OPEN_ARGS_CLAIM_FH            = 4;
   ///    OPEN_ARGS_CLAIM_DELEG_CUR_FH  = 5;
   ///    OPEN_ARGS_CLAIM_DELEG_PREV_FH = 6;
   /// };
   ///

   ///
   /// enum open_args_share_access4 = {
   ///    OPEN_ARGS_CREATE_MODE_GUARDED    = 0;
   ///    OPEN_ARGS_CREATE_MODE_EXCLUSIVE  = 1;
   /// };
   ///

   ///
   /// typedef open_arguments4 fattr4_open_arguments;
   ///

   ///
   /// %/*
   /// % * Determine what OPEN4 supports.
   /// % */
   /// const FATTR4_OPEN_ARGUMENTS     = 86;
   ///

   ///
   /// const OPEN4_SHARE_ACCESS_WANT_OPEN_XOR_DELEGATION = 0x200000;
   ///
```

```
///
/// const OPEN4_RESULT_NO_OPEN_STATEID = 0x00000010;
///
```

<CODE ENDS>

[RFC8178] (see Section 4.4.2) allows for extending the microversion of the NFSv4.x protocol without increasing the microversion. The client can probe the capabilities of the server and based on that result, determine if both it and the server support features not specified in the main microversion documuent.

The XDR extensions presented in this section allow for the OPEN procedure to be extended in such a fashion. It models all of the parameters via bitmap4 data structures, which allows for the addition of a new flag to any of the OPEN arguments (see Section 18.16.1 of [RFC5661]. Two new flags are provided:

- OPEN4_SHARE_ACCESS_WANT_OPEN_XOR_DELEGATION (see Section Section 4)

- OPEN4_SHARE_ACCESS_WANT_DELEG_TIMESTAMPS

Subsequent documents can use this framework to introduce new functionality to OPEN.

This section extends the OPEN procedure to allow it to return either an open or delegation stateid. The file is said to be "open" to the client as long as the count of open and delegated stateids is greater than 0. Either type of stateid is suffucient to keep the file open, which allows READ, WRITE, LOCK, and LAYOUTGET operations to proceed. If the client gets both a open and a delegation stateid as part of the OPEN, then it has to return them both. And during each operation, the client can send a costly GETATTR.

If the client knows that the server supports the OPEN4_SHARE_ACCESS_WANT_OPEN_XOR_DELEGATION flag (as determined by an earlier GETATTR operation which queried for the FATTR4_OPEN_ARGUMENTS attribute), then the client can supply that flag during the OPEN and only get either an open or delegation stateid.

The client is already prepared to not get a delegation stateid even if requested. In order to not send an open stateid, the server can indicate that fact with the result flag of OPEN4_RESULT_NO_OPEN_STATEID. The open stateid field, OPEN4resok.stateid (see Section 18.16.2 of [RFC5661]), should also be set to the special all zero stateid.

3. Offline Files

```
<CODE BEGINS>

///
/// typedef bool            fattr4_offline;
///

///
/// const FATTR4_OFFLINE            = 83;
///

<CODE ENDS>
```

If a file is archived or offline, then the metadata portion is available to the file server, but the data content is not available. Thus a compound with a GETATTR or READDIR can report the file's attributes without bringing the file online. However, either an OPEN or a LAYOUTGET might cause the file server to retrieve the archived data contents, bringing the file online. If an operating system is not aware that the file is offline, it might inadvertantly open the file to determine what type of file it is accessing. [[AI2: Document these OSes! --TH]]

By adding the new attribute FATTR4_OFFLINE, a client can predetermine the avaiablity of the file, avoiding the need to open it at all. Note that being offline might also mean that the file is archived in the cloud, i.e., there can be an expense in both retrieving the file to bring online and in sending the file back to offline status.

4. Proxying of Times

```
<CODE BEGINS>

///
/// /*
///  * attributes for the delegation times being
///  * cached and served by the "client"
///  */
/// typedef nfstime4        fattr4_time_deleg_access;
/// typedef nfstime4        fattr4_time_deleg_modify;
///
```

```
///
/// %/*
/// % * New RECOMMENDED Attribute for
/// % * delegation caching of times
/// % */
/// const FATTR4_TIME_DELEG_ACCESS  = 84;
/// const FATTR4_TIME_DELEG_MODIFY  = 85;
///

///
/// const OPEN4_SHARE_ACCESS_WANT_DELEG_TIMESTAMPS = 0x100000;
///
```

<CODE ENDS>

When a client is granted a write delegation on a file, it is the authority for the file. If the server queries the client as to the state of the file via a CB_GETATTR (see Section 20.1 of [RFC5661]), then it can only determine the size of the file. Likewise, if the client holding the delegation wants to know either of the access, modify, or change times, it has to send a GETATTR to the server. While it is the authority for these values, it has no way to guarantee these values after the delegation has been returned. And as such, it can not pass these times up to an application expecting posix compliance. [[AI3: Cite --TH]]

With the addition of the new flag: OPEN4_SHARE_ACCESS_WANT_DELEG_TIMESTAMPS, the client and server can negiotiate that the client will be the authority for these values and upon return of the delegation stateid via a DELEGRETURN (see section 18.6 of [RFC5661]), the times will be passed back to the server. If the server is queried by another client for either the size or the times, it will need to use a CB_GETATTR to query the client which holds the delegation (see Section 20.1 of [RFC5661]).

If a server informs the client via the FATTR4_OPEN_ARGUMENTS attribute that it supports OPEN_ARGS_SHARE_ACCESS_WANT_DELEG_TIMESTAMPS and it returns a valid delegation stateid for an OPEN operation which sets the OPEN4_SHARE_ACCESS_WANT_DELEG_TIMESTAMPS flag, then it MUST be able to query the client via a CB_GETATTR for the FATTR4_TIME_DELEG_ACCESS attribute and FATTR4_TIME_DELEG_MODIFY attribute. (The change time can be derived from the modify time.) Further, when it gets a SETATTR (see Section 18.30 of [RFC5661]) in the same compound as the DELEGRETURN, then it MUST accept those FATTR4_TIME_DELEG_ACCESS attribute and FATTR4_TIME_DELEG_MODIFY attribute changes and derive the change time or reject the changes with NFS4ERR_DELAY.

A key prequisite of this approach is that the server and client are in time synchronization with each other. Note that while the base NFSv4.2 does not require such synchronization, the use of RPCSEC_GSS typically makes such a requirement. When the client presents either FATTR4_TIME_DELEG_ACCESS or FATTR4_TIME_DELEG_MODIFY attributes to the server, the server MUST decide whether the times presented are before the old times or past the current time. If the time presented is before the original time, then the update is ignored. If the time presented is in the future, the server can either clamp the new time to the current time, or it may return NFS4ERR_DELAY to the client, allowing it to retry. Note that if the clock skew is large, this policy will result in access to the file being denied until such time that the clock skew is exceeded.

A change in the access time MUST not advance the change time, also known as the time_metadata attribute (see Section 5.8.2.42 of [RFC5661]), but a change in the modify time might advance the change time (and in turn the change attribute (See Section 5.8.1.4 of [RFC5661]). If the modify time is greater than the change time and before the current time, then the change time is adjusted to the modify time and not the current time (as is most likely done on most SETATTR calls that change the metadata). If the modify time is in the future, it will be clamped to the current time.

Note that each of the possible times, access, modify, and change, are compared to the current time. They should all be compared against the same time value for the current time. I.e., do not retrieve a different value of the current time for each calculation.

If the client sets the OPEN4_SHARE_ACCESS_WANT_DELEG_TIMESTAMPS flag in an OPEN operation, then it MUST support the FATTR4_TIME_DELEG_ACCESS and FATTR4_TIME_DELEG_MODIFY attributes both in the CB_GETATTR and SETATTR operations.

4.1. Use case

With a server is a proxy for a NFSv4 server, it is a client to the NFSv4 server and during file I/O, it may get a delegation on a file. The client of the proxy would be querying the proxy for attributes and not the NFSv4 server. Each GETATTR from that client would result in at least one additional GETATTR being sent across the wire.

5. XDR Description of the Flexible File Layout Type

This document contains the external data representation (XDR) [RFC4506] description of the new open flags for delegating the file to the client. The XDR description is embedded in this document in a way that makes it simple for the reader to extract into a ready-to-

compile form. The reader can feed this document into the following shell script to produce the machine readable XDR description of the new flags:

<CODE BEGINS>

```
#!/bin/sh
grep '^ *///' $* | sed 's?^ */// ??' | sed 's?^ *///$??'
```

<CODE ENDS>

That is, if the above script is stored in a file called "extract.sh", and this document is in a file called "spec.txt", then the reader can do:

```
sh extract.sh < spec.txt > delstid_prot.x
```

The effect of the script is to remove leading white space from each line, plus a sentinel sequence of "///". XDR descriptions with the sentinel sequence are embedded throughout the document.

Note that the XDR code contained in this document depends on types from the NFSv4.2 nfs4_prot.x file (generated from [RFC7863]). This includes both nfs types that end with a 4, such as offset4, length4, etc., as well as more generic types such as uint32_t and uint64_t.

While the XDR can be appeneded to that from [RFC7863], the various code snippets belong in their respective areas of the that XDR.

5.1. Code Components Licensing Notice

Both the XDR description and the scripts used for extracting the XDR description are Code Components as described in Section 4 of "Legal Provisions Relating to IETF Documents" [LEGAL]. These Code Components are licensed according to the terms of that document.

6. Security Considerations

There are no new security considerations beyond those in [RFC7862].

7. IANA Considerations

There are no IANA considerations.

8. Normative References

[LEGAL] IETF Trust, "Legal Provisions Relating to IETF Documents", November 2008, <http://trustee.ietf.org/docs/IETF-Trust-License-Policy.pdf>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4506] Eisler, M., "XDR: External Data Representation Standard", STD 67, RFC 4506, May 2006.

[RFC5661] Shepler, S., Ed., Eisler, M., Ed., and D. Noveck, Ed., "Network File System (NFS) Version 4 Minor Version 1 Protocol", RFC 5661, January 2010.

[RFC7862] Haynes, T., "NFS Version 4 Minor Version 2", RFC 7862, November 2016.

[RFC7863] Haynes, T., "Network File System (NFS) Version 4 Minor Version 2 External Data Representation Standard (XDR) Description", RFC 7863, November 2016.

[RFC8178] Noveck, D., "Rules for NFSv4 Extensions and Minor Versions", RFC 8178, July 2017.

Appendix A. Acknowledgments

Trond Myklebust and David Flynn all worked on the prototype at Hammerspace.

Appendix B. RFC Editor Notes

[RFC Editor: please remove this section prior to publishing this document as an RFC]

[RFC Editor: prior to publishing this document as an RFC, please replace all occurrences of RFCTBD10 with RFCxxxx where xxxx is the RFC number of this document]

Author's Address

Thomas Haynes
Hammerspace
4300 El Camino Real Ste 105
Los Altos, CA 94022
USA

Email: loghyr@hammer.space