Network File System Version 4
Internet-Draft
Intended status: Standards Track
Expires: 22 May 2026

T. Haynes
Hammerspace

18 November 2025

Adding an Uncacheable File Attribute to NFSv4.2
draft-ietf-nfsv4-uncacheable-files-00

Abstract

The Network File System version 4.2 (NFSv4.2) allows a client to cache data for file objects. Caching file data can lead to performance issues if the cache hit rate is low. This document introduces a new uncacheable file attribute for NFSv4.2. Files marked as uncacheable MUST NOT be stored in client-side caches. This document extends NFSv4.2 (see RFC7862).

Note to Readers

Discussion of this draft takes place on the NFSv4 working group mailing list (nfsv4@ietf.org), which is archived at https://mailarchive.ietf.org/arch/search/?email_list=nfsv4. Source code and issues list for this draft can be found at https://github.com/ietf-wg-nfsv4/uncacheable-files.

Working Group information can be found at https://github.com/ietf-wg-nfsv4.

Status of This Memo

Table of Contents

1. Introduction

With a remote filesystem, the client typically caches file contents in order to improve performance. Several assumptions are made about the rate of change in the number of clients trying to concurrently access a file. With NFSv4.2, this could practically be mitigated practically by file delegations for the file contents.

There are prior efforts to bypass file caching. In Highly Parallel Computing (HPC) workloads, file caching is bypassed in order to achieve consistent work flows.

This document introduces the uncacheable file attribute to NFSv4.2 to bypass file caching on the client. As such, it is an OPTIONAL to implement attribute for NFSv4.2. However, if both the client and the server support this attribute, then the client MUST follow the semantics of uncacheable.
// What about mixed modes?

A client can easily determine whether or not a server supports the uncacheable file attribute with a simple GETATTR on any file.
// Is this still true? If the server does not support the uncacheable attribute, it will return an error of NFS4ERR_ATTRNOTSUPP.

The only way that the server can determine that the client supports the attribute is if the client sends either a GETATTR or a SETATTR with the uncacheable attribute.

As bypassing file caching is file based, it is only applicable for dirents which are of type attribute value of NF4REG.

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

file caching A client cache, normally called the page cache, which caches the contents of a regular file. Typical usage would be to accumulate changes to be bunched together for writing to the server.

Further, the definitions of the following terms are referenced as follows:

* file delegations (Section 10.2 of [RFC8881])

* GETATTR (Section 18.7 of [RFC8881])

* NF4REG (Section 5.8.1.2 of [RFC8881])

* NFS4ERR_ATTRNOTSUPP (Section 15.1.15.1 of [RFC8881]

* SETATTR (Section 18.30 of [RFC8881])

* system (Section 5.8.2.36 of [RFC8881])

1.2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Caching of File Data

The uncacheable file attribute instructs the client to bypass its page cache for the file. This behavior is similar to using the O_DIRECT flag with the open call ([open]). This can be beneficial for files that are not shared or for files that do not exhibit access patterns suitable for caching.

However, the real need for bypassing write caching is evident in HPC workloads. In general, these involve massive data transfers and require extremely low latency. Write caching can introduce unpredictable latency, as data is buffered and flushed later.

2.1. Uncacheable Files

If a file object is marked as uncacheable, all modifications to the file MUST be immediately sent from the client to the server. In other words, the file data is also not cacheable.

3. XDR for Offline Attribute

```
///
/// typedef bool            fattr4_uncacheable_file;
///
/// const FATTR4_UNCACHEABLE_FILE       = 87;
///
```

4. Extraction of XDR

This document contains the external data representation (XDR) [RFC4506] description of the uncacheable attribute. The XDR description is presented in a manner that facilitates easy extraction into a ready-to-compile format. To extract the machine-readable XDR description, use the following shell script:

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

For example, if the script is named 'extract.sh' and this document is named 'spec.txt', execute the following command:

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

This script removes leading blank spaces and the sentinel sequence '////' from each line. 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 appended to that from [RFC7863], the code snippets should be placed in their appropriate sections within the existing XDR.

5. Security Considerations

This document imposes no new security considerations to NFSv4.2.

6. IANA Considerations

This document has no IANA actions.

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC4506] Eisler, M., Ed., "XDR: External Data Representation Standard", STD 67, RFC 4506, DOI 10.17487/RFC4506, May 2006, <https://www.rfc-editor.org/rfc/rfc4506>.

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2", FYI 36, RFC 4949, DOI 10.17487/RFC4949, August 2007, <https://www.rfc-editor.org/rfc/rfc4949>.

[RFC7862] Haynes, T., "Network File System (NFS) Version 4 Minor Version 2 Protocol", RFC 7862, DOI 10.17487/RFC7862, November 2016, <https://www.rfc-editor.org/rfc/rfc7862>.

[RFC7863] Haynes, T., "Network File System (NFS) Version 4 Minor Version 2 External Data Representation Standard (XDR) Description", RFC 7863, DOI 10.17487/RFC7863, November 2016, <https://www.rfc-editor.org/rfc/rfc7863>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC8178] Noveck, D., "Rules for NFSv4 Extensions and Minor Versions", RFC 8178, DOI 10.17487/RFC8178, July 2017, <https://www.rfc-editor.org/rfc/rfc8178>.

[RFC8881] Noveck, D., Ed. and C. Lever, "Network File System (NFS) Version 4 Minor Version 1 Protocol", RFC 8881, DOI 10.17487/RFC8881, August 2020, <https://www.rfc-editor.org/rfc/rfc8881>.

7.2. Informative References

[open] "open and create files.", Linux Programmer’s Manual , n.d..

[POSIX.1] IEEE, "The Open Group Base Specifications Issue 7", IEEE Std 1003.1, 2013 Edition , 2013.

[RFC1813] Callaghan, B., Pawlowski, B., and P. Staubach, "NFS Version 3 Protocol Specification", RFC 1813, DOI 10.17487/RFC1813, June 1995, <https://www.rfc-editor.org/rfc/rfc1813>.

Acknowledgments

Trond Myklebust, Mike Snitzer, and Thomas Haynes all worked on the prototype at Hammerspace.

Chris Inacio, Brian Pawlowski, and Gorry Fairhurst helped guide this process.

Author’s Address

Thomas Haynes
Hammerspace
Email: loghyr@gmail.com