Workgroup: Network File System Version 4
Internet-Draft: draft-ietf-nfsv4-layoutwcc-00
Updates: 8435 (if approved)
Published: 21 February 2023
Intended Status: Standards Track
Expires: 25 August 2023
Authors: T. Haynes   T. Myklebust
Hammerspace   Hammerspace

Add LAYOUT_WCC to NFSv4.2's Flex File Layout Type

Abstract

The Parallel Network File System (pNFS) Flexible File Layout allows for a file's metadata (MDS) and data (DS) to be on different servers. It does not provide a mechanism for the data server to update the metadata server of changes to the data part of the file. The client has knowledge of such updates, but lacks the ability to update the metadata server. This document presents a refinement to RFC8435 to allow the client to update the metadata server to changes on the data server.

This note is to be removed before publishing as an RFC.

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/browse/nfsv4/. Working Group information can be found at https://datatracker.ietf.org/wg/nfsv4/about/.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

In the Network File System version4 (NFSv4) with a Parallel NFS (pNFS) Flexible File Layout ([RFC8435]) server, there is no mechanism for the data servers to update the metadata servers for when the data portion of the file is modified. The metadata server needs this knowledge to correspondingly update the metadata portion of the file. If the client is using NFSv3 as the protocol with the data server, it can leverage weak cache consistency (WCC) to update the metadata server of the attribute changes. In this document, we introduce a new operation called LAYOUT_WCC which allows the client to periodically report the attributes of the data files to the metadata server.

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) data: that part of the file system object that contains the data to be read or written. It is the contents of the object rather than the attributes of the object.

data server (DS): a pNFS server that provides the file's data when the file system object is accessed over a file-based protocol.

(file) metadata: the part of the file system object that contains various descriptive data relevant to the file object, as opposed to the file data itself. This could include the time of last modification, access time, EOF position, etc.

metadata server (MDS): the pNFS server that provides metadata information for a file system object.

weak cache consistency (WCC): In NFSv3, WCC allows the client to check for file attribute changes before and after an operation. (See Section 2.6 of [RFC1813])

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. Operation 77: LAYOUT_WCC - Layout Weak Cache Consistency

2.1. ARGUMENT

```
<CODE BEGINS>
/// struct LAYOUT_WCC4args {
///         stateid4        lowa_stateid;
///         layouttype4     lowa_type;
///         opaque          lowa_body<>;
/// };

<CODE ENDS>
```

2.2. RESULT

```
<CODE BEGINS>
/// struct LAYOUT_WCC4res {
///         nfsstat4                lowr_status;
/// };

<CODE ENDS>
```

2.3. DESCRIPTION

When using pNFS (See Section 12 of [RFC8881]), the client is most likely to be performing file operations to the storage device and not the metadata server. With a NFSv3 data server in the flexible files layout type (in [RFC8435]) there is no control protocol ([RFC8434]) between the metadata server and the storage device. In order to update the metadata state of the file, the metadata server will need to track the metadata state of the data file - once the layout is issued, it is not able to see the NFSv3 file operations from the client to the storage device. Thus the metadata server will be required to query the storage device for the data file attributes.

For example, the metadata server would issue a NFSv3 GETATTR to the storage device. These queries are most likely triggered in response to a NFSv4 GETATTR to the metadata server. Not only are these NFSv3 GETATTRs to the storage device individually expensive, the storage device can become inundated by a storm of such requests. NFSv3 solved a similar issue by having the READ and WRITE operations employ a post-operation attribute to report the weak cache consistency (WCC) data (See Section 2.6 of [RFC1813]).

Each NFSv3 operation corresponds to one round trip between the client and server. So a WRITE followed by a GETATTR would require two round trips. In that scenario, the attribute information retrieved is considered to be strict server-client consistency. For NFSv4, the WRITE and GETATTR can be issued together inside a compound, which only requires one round trip between the client and server. And this is also considered to be a strict server-client consistency. In essence, the NFSv4 READ and WRITE operations drop the post-operation attributes, allowing the client to decide if it needs that information.

With the flexible files layout type, the client can leverage the NFSv3 WCC to service the proxying of times (See Section 4 of [delstid]). But the granularity of this data is limited. With client side mirroring (See Section 8 of [RFC8435]), the client has to aggregate the N mirrored files in order to send one piece of information instead of N pieces of information. Also, the client is limited to sending that information only when it returns the delegation.

The current filehandle and the lowa_stateid identifies the particular layout for the LAYOUT_WCC operation. The lowa_type indicates how to unpack the layout type specific payload inside the lowa_body field. The lowa_type is defined to be a value from the IANA registry for "pNFS Layout Types Registry".

The lowa_body will contain the data file attributes. The client will be responsible for mapping the NFSv3 post-operation attributes to those in a fattr4. Just as the post-operation attributes may be ignored by the client, the server may ignore the attributes inside the LAYOUT_WCC. But the server can also use those attributes to avoid querying the storage device for the data file attributes. Note that as these attributes are optional and there is nothing the client can do if the server ignores one, there is no need to return a bitmap4 of which attributes were accepted in the result of the LAYOUT_WCC.

2.4. Allowed Errors

The LAYOUT_WCC operation can raise the errors in Table 1. When an error is encountered, the metadata server can decide to ignore the entire operation or depending on the layout type specific payload, it could decide to apply a portion of the payload.

Valid Error Returns for LAYOUT_WCC

Errors
NFS4ERR_ADMIN_REVOKED, NFS4ERR_BADXDR, NFS4ERR_BAD_STATEID, NFS4ERR_DEADSESSION, NFS4ERR_DELAY, NFS4ERR_DELEG_REVOKED, NFS4ERR_EXPIRED, NFS4ERR_FHEXPIRED, NFS4ERR_GRACE, NFS4ERR_INVAL, NFS4ERR_ISDIR, NFS4ERR_MOVED, NFS4ERR_NOFILEHANDLE, NFS4ERR_NOTSUPP, NFS4ERR_NO_GRACE, NFS4ERR_OLD_STATEID, NFS4ERR_OP_NOT_IN_SESSION, NFS4ERR_REP_TOO_BIG, NFS4ERR_REP_TOO_BIG_TO_CACHE, NFS4ERR_REQ_TOO_BIG, NFS4ERR_RETRY_UNCACHED_REP, NFS4ERR_SERVERFAULT, NFS4ERR_STALE, NFS4ERR_TOO_MANY_OPS, NFS4ERR_UNKNOWN_LAYOUTTYPE, NFS4ERR_WRONG_CRED, NFS4ERR_WRONG_TYPE

Table 1

2.5. Extension of Existing Implementations

The new LAYOUT_WCC operation is **OPTIONAL** for both NFSv4.2 ([RFC7863]) and the flexible files layout type ([RFC8435]).

2.6. Flex Files Layout Type

```
<CODE BEGINS>
/// struct ff_data_server_wcc4 {
///             deviceid4           ffdsw_deviceid;
///             stateid4            ffdsw_stateid;
///             nfs_fh4             ffdsw_fh_vers<>;
///             fattr4              ffdsw_attributes;
/// };
///
/// struct ff_mirror_wcc4 {
///             ff_data_server_wcc4  ffmw_data_servers<>;
/// };
///
/// struct ff_layout_wcc4 {
///             ff_mirror_wcc4      fflw_mirrors<>;
/// };


<CODE ENDS>
```

The flex file layout type specific results **MUST** correspond to the ff_layout4 data structure as defined in Section 5.1 of [RFC8435]. There **MUST** be a one-to-one correspondence between:

*ff_data_server4 -> ff_data_server_wcc4

*ff_mirror4 -> ff_mirror_wcc4

*ff_layout4 -> ff_layout_wcc4

Each ff_layout4 has an array of ff_mirror4, which have an array of ff_data_server4. Based on the current filehandle and the lowa_stateid, the server can match the reported attributes.

But the positional correspondence between the elements is not sufficient to determine the attributes to update. Consider the case where a layout had three mirrors and two of them had updated attributes, but the third did not. A client could decide to present all three mirrors, with one mirror having an attribute mask with no attributes present. Or it could decide to present only the two mirrors which had been changed.

In either case, the combination of ffdsw_deviceid, ffdsw_stateid, and ffdsw_fh_vers will uniquely identify the attributes to be updated. All three arguments are required. A layout might have multiple data files on the same storage device, in which case the ffdsw_deviceid and ffdsw_stateid would match, but the ffdsw_fh_vers would not.

The ffdsw_attributes are processed similar to the obj_attributes in the SETATTR arguments (See Section 18.34 of [RFC8881]).

3. Extraction of XDR

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:

```
<CODE BEGINS>
sh extract.sh < spec.txt > layout_wcc.x


<CODE ENDS>
```

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 appended to that from [RFC7863], the various code snippets belong in their respective areas of the that XDR.

3.1. 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.

4. Security Considerations

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

5. IANA Considerations

IANA should use the current document (RFC-TBD) as the reference for the new entries.

6. References

6.1. Normative References

[delstid] Haynes, T. and T. Myklebust, "Extending the Opening of Files in NFSv4.2", draft-ietf-nfsv4-delstid-02.xml (Work In Progress), February 2023.

[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/info/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/info/rfc4506>.

[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/info/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/info/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/info/rfc8174>.

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

[RFC8434] Haynes, T., "Requirements for Parallel NFS (pNFS) Layout Types", RFC 8434, DOI 10.17487/RFC8434, August 2018, <https://www.rfc-editor.org/info/rfc8434>.

[RFC8435] Halevy, B. and T. Haynes, "Parallel NFS (pNFS) Flexible File Layout", RFC 8435, DOI 10.17487/RFC8435, August 2018, <https://www.rfc-editor.org/info/rfc8435>.

[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/info/rfc8881>.

6.2. Informative References

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

[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/info/rfc1813>.

Appendix A. Acknowledgments

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

Dave Noveck and Tigran Mkrtchyan provided reviews of the document.

Authors' Addresses

Thomas Haynes
Hammerspace

Email: loghyr@hammerspace.com

Trond Myklebust
Hammerspace

Email: trondmy@hammerspace.com