Stream: Internet Engineering Task Force (IETF)
RFC: 9737
Category: Standards Track
Published: February 2025
ISSN: 2070-1721
Authors: T. Haynes, *Hammerspace*; T. Myklebust, *Hammerspace*

RFC 9737
Reporting of Errors via LAYOUTRETURN in NFSv4.2

Abstract

The Parallel Network File System (pNFS) allows for a file's metadata (MDS) and data (DS) to be on different servers. When the metadata server is restarted, the client can still modify the data file component. During the recovery phase of startup, the metadata server and the data servers work together to recover state (which files are open, last modification time, size, etc.). If the client has not encountered errors with the data files, then the state can be recovered and the resilvering of the data files can be avoided. With any errors, there is no means by which the client can report errors to the metadata server. As such, the metadata server has to assume that a file needs resilvering. This document presents an extension to RFC 8435 to allow the client to update the metadata and avoid resilvering.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9737.

Copyright Notice

Table of Contents

1. Introduction

In the Network File System version 4 (NFSv4) with a Parallel NFS (pNFS) Flexible File Layout [RFC8435] server, during recovery after a restart, there is no mechanism for the client to inform the metadata server about an error that occurred during a WRITE operation (see Section 18.32 of [RFC8881]) to the data servers in the period of the outage.

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

See Section 1.1 of [RFC8435] for a set of definitions.

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. Layout State Recovery

When a metadata server restarts, clients are provided a grace recovery period where they are allowed to recover any state that they had established. With open files, the client can send an OPEN operation (see Section 18.16 of [RFC8881]) with a claim type of CLAIM_PREVIOUS (see Section 9.11 of [RFC8881]). The client uses the RECLAIM_COMPLETE operation (see Section 18.51 of [RFC8881]) to notify the metadata server that it is done reclaiming state.

The NFSv4 Flexible File Layout Type allows for the client to mirror files (see Section 8 of [RFC8435]). With client-side mirroring, it is important for the client to inform the metadata server of any I/O errors encountered with one of the mirrors. This is the only way for the metadata server to determine if one or more of the mirrors are corrupt and then repair the mirrors via resilvering (see Section 1.1 of [RFC8435]). The client can use LAYOUTRETURN (see Section 18.44 of [RFC8881]) and the ff_ioerr4 structure (see Section 9.1.1 of [RFC8435]) to inform the metadata server of I/O errors.

A problem arises when the metadata server restarts and the client has errors it needs to report but cannot do so. Section 12.7.4 of [RFC8881] requires that the client **MUST** stop using layouts. While the intent there is that the client **MUST** stop doing I/O to the storage devices, it is also true that the layout stateids are no longer valid. The LAYOUTRETURN needs a layout stateid to proceed, and the client cannot get a layout during grace recovery (see Section 12.7.4 of [RFC8881]) to recover layout state. As such, clients have no choice but to not recover files with I/O errors. In turn, the metadata server **MUST** assume that the mirrors are inconsistent and pick one for resilvering. It is a **MUST** because even if the metadata server can determine that the client did modify data during the outage, it **MUST NOT** assume those modifications were consistent.

To fix this issue, the metadata server **MUST** accept the anonymous stateid of all zeros (see Section 8.2.3 of [RFC8881]) for the lrf_stateid in LAYOUTRETURN (see Section 18.44.1 of [RFC8881]). The client can use this anonymous stateid to inform the metadata server of errors encountered. The metadata server can then accurately resilver the file by picking the mirror(s) that does not have any associated errors.

During the grace period, if the client sends an lrf_stateid in the LAYOUTRETURN with any value other than the anonymous stateid of all zeros, then the metadata server **MUST** respond with an error of NFS4ERR_GRACE (see Section 15.1.9.2 of [RFC8881]). After the grace period, if the client sends an lrf_stateid in the LAYOUTRETURN with a value of the anonymous stateid of all zeros, then the metadata server **MUST** respond with an error of NFS4ERR_NO_GRACE (see Section 15.1.9.3 of [RFC8881]).

Also, when the metadata server builds the reply to the LAYOUTRETURN when an lrf_stateid with the value of the anonymous stateid of all zeros it **MUST NOT** bump the seqid of the lorr_stateid.

If the metadata server detects that the layout being returned in the LAYOUTRETURN does not match the current mirror instances found for the file, then it **MUST** ignore the LAYOUTRETURN and resilver the file in question.

The metadata server **MUST** resilver any files that are neither explicitly recovered with a CLAIM_PREVIOUS nor have a reported error via a LAYOUTRETURN. The client has most likely restarted and lost any state.

2.1. When to Resilver

A write intent occurs when a client opens a file and gets a LAYOUTIOMODE4_RW from the metadata server. The metadata server **MUST** track outstanding write intents, and when it restarts, it **MUST** track recovery of those write intents. The method that the metadata server uses to track write intents is implementation specific, i.e., outside the scope of this document.

The decision to resilver a file depends on how the client recovers the file before the grace period ends. If the client reclaims the file and reports no errors, the metadata server **MUST NOT** resilver the file. If the client reports an error on the file, then the file **MUST** be resilvered. If the client does not reclaim or report an error before the grace period ends, then under the old behavior, the metadata server **MUST** resilver the file.

The resilvering process is broadly to:

1. fence the file (see Section 2.2 of [RFC8435]),
2. record the need to resilver,
3. release the write intent, and
4. once there are no write intents on the file, start the resilvering process.

The metadata server **MUST NOT** resilver a file if there are clients with outstanding write intents, i.e., multiple clients might have the file open with write intents. As the metadata server **MUST** track write intents, it **MUST** also track the need to resilver, i.e., if the metadata server restarts during the grace period, it **MUST** restart the file recovery if it replays the write intent, or else it **MUST** start the resilvering if it replays the resilvering intent.

Whether the metadata server prevents all I/O to the file until the resilvering is done, forces all I/O to go through the metadata server, or allows a proxy server to update the new data file as it is being resilvered is all an implementation choice. The constraint is that the metadata server is responsible for the reconstruction of the data file and for the consistency of the mirrors.

If the metadata server does allow the client access to the file during the resilvering, then the client **MUST** have the same layout (set of mirror instances) after the metadata server as before. One way that such a resilvering can occur is for a proxy server to be inserted into the layout.

That server will be copying a good mirror instance to a new instance. As it gets I/O via the layout, it will be responsible for updating the copy it is performing. This requirement is that the proxy server **MUST** stay in the layout until the grace period is finished.

2.2. Version Mismatch Considerations

The metadata server has no expectations for the client to use this new functionality. Therefore, if the client does not use it, the metadata server will function normally.

If the client does use the new functionality and the metadata server does not support it, then the metadata server **MUST** reply with a NFS4ERR_BAD_STATEID to the LAYOUTRETURN. If the client detects a NFS4ERR_BAD_STATEID error in this scenario, it should fall back to the old behavior of not reporting errors.

3. Security Considerations

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

4. IANA Considerations

This document has no IANA actions.

5. References

5.1. 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/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>.

[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>.

Acknowledgments

Tigran Mkrtchyan, Jeff Layton, and Rick Macklem provided reviews of the document.

Authors' Addresses

Thomas Haynes
Hammerspace
Email: loghyr@gmail.com

Trond Myklebust
Hammerspace
Email: trondmy@hammerspace.com