NFSv4 Working Group
Internet-Draft
Intended status: Proposed Standard
Expires: April 14, 2011
Updates: 5661, 5662

S. Faibish
EMC Corporation
D. Black
EMC Corporation
M. Eisler
NetApp
J. Glasgow
Google

October 14, 2010

pNFS Access Permissions Check
draft-ietf-nfsv4-pnfs-access-permissions-check-00

Abstract

This document extends the pNFS protocol to communicate errors caused by inability to access data servers referenced by layouts, including checks performed by both clients and the MDS. The extension provides means for clients to communicate client-detected access denial errors to the MDS, including the case in which a client requests direct NFS access via the MDS that the MDS cannot perform.

Table of Contents

1. Introduction

Figure 1 shows the overall architecture of a Parallel NFS (pNFS) system:

```
          +-----------+
          |+-----------+                                 +-----------+
          ||+-----------+                                |           |
          |||           |       NFSv4.1 + pNFS           |           |
          +||  Clients  |<------------------------------>|    MDS    |
           +|           |                                |           |
            +-----------+                                |           |
                 |||                                     +-----------+
                 |||                                           |
                 |||                                           |
                 ||| Storage        +-----------+              |
                 ||| Protocol       |+-----------+             |
                 ||+----------------||+-----------+  Control   |
                 |+-----------------|||           |  Protocol  |
                 +------------------+||  Storage  |------------+
                                     +|  Devices  |
                                      +-----------+
```

Figure 1 pNFS Architecture

In this document, "storage device" is used as a general term for a data server and/or storage server for the file, block or object pNFS layouts.

The current pNFS protocol [RFC5661] assumes that a client can access every storage device (SD) included in a valid layout sent by the MDS server, and provides no means to communicate client access failures to the MDS. Access failures can impair pNFS performance scaling and allow significant errors to go unreported. If the MDS can access all the storage devices involved, but the client doesn't have sufficient access rights to some storage devices, the client may choose to fall back to accessing the file system using NFSV4.1 without pNFS support; there are environments in which this behavior is undesirable, especially if it occurs silently. An important example is addition of a new storage device to which a large population of pNFS clients (e.g., 1000s) lacks access permission. Layouts granted that use this new device, result in client errors, requiring that all I/Os to that new storage device be served by the MDS server. This creates a performance and scalability bottleneck that may be difficult to detect based on I/O behavior because the other storage devices are functioning correctly.

The preferable approach to this scenario is to report the access failures before any client attempts to issue any I/Os that can only be serviced by the MDS server. This makes the problem explicit, rather than forcing the MDS, or a system administrator, to diagnose the performance problem caused by client I/O using NFS instead of pNFS. There are limits to this approach because complex mount structures may prevent a client from detecting this situation at mount time, but at a minimum, access problems involving the root of the mount structure can be detected.

The most suitable time for the client to report inability to access a storage device is at mount time, but this is not always possible. If the application uses a special tag or a switch to the mount command (e.g., -pnfs) and syscall to declare its intention to use pNFS, at the client, the client can check for both pNFS support and device accessibility.

This document introduces an error reporting mechanism that is an extension to the return of a pNFS layout; a pNFS client MAY use this mechanism to inform the MDS that the layout is being returned because one or more data servers are not accessible to the client. Error reporting at I/O time is not affected because the result of an inaccessible data server may not be an I/O error if a subsequent retry of the operation via the MDS is successful.

There is a related problem scenario involving an MDS that cannot access some storage devices and hence cannot perform I/Os on behalf of a client. In the case of the block layout [RFC5663] if the MDS lacks access to a storage device (e.g., LUN), MDS implementations generally do not export any filesystem using that storage device. In contrast to the block layout, MDSs for the file [RFC5661] and object [RFC5664] layouts may be unable to access the storage devices that store data for an exported filesystem. This enables a file or object layout MDS to provide layouts that contain client-inaccessible devices. For the specific case of adding a new storage device to a filesystem, MDS issuance of test I/Os to the newly added device before using it in layouts avoids this problem scenario, but does not cover loss of access to existing storage devices at a later time.

In addition, [RFC5661] states that a client can write through or read from the MDS, even if it has a layout; this assumes that the MDS can access all the storage devices. This document makes that assumed access an explicit requirement.

2. Conventions used in this document

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 RFC-2119 [RFC2119].

3. Changes to Operation 51: LAYOUTRETURN (RFC 5661)

The existing LAYOUTRETURN operation is extended by introducing three new layout return types that correspond to the existing types:

- o LAYOUT4_RET_REC_FILE_NO_ACCESS at file scope;
- o LAYOUT4_RET_REC_FSID_NO_ACCESS at fsid scope; and
- o LAYOUT4_RET_REC_ALL_NO_ACCESS at client scope.

The first return type returns the layout for an individual file and informs the server that the reason for the return is a storage device connectivity problem. The second return type performs that function for all layouts held by the client for the filesystem that corresponds to the current filehandle used for the LAYOUTRETURN operation. The third return type performs that function for all layouts held by the client; it is intended for situations in which a device is shared across all or most of the filesystems from a server for which the client has layouts.

3.1. ARGUMENT (18.44.1)

The ARGUMENT specification of the LAYOUTRETURN operation in section 18.44.1 of [RFC5661] is replaced by the following XDR code [XDR]:

```
/* Constants used for new LAYOUTRETURN and CB_LAYOUTRECALL */
const LAYOUT4_RET_REC_FILE      = 1;
const LAYOUT4_RET_REC_FSID      = 2;
const LAYOUT4_RET_REC_ALL       = 3;
const LAYOUT4_RET_REC_FILE_NO_ACCESS    = 4;
const LAYOUT4_RET_REC_FSID_NO_ACESSS    = 5;
const LAYOUT4_RET_REC_ALL_NO_ACCESS     = 6;

enum layoutreturn_type4 {
     LAYOUTRETURN4_FILE = LAYOUT4_RET_REC_FILE,
     LAYOUTRETURN4_FSID = LAYOUT4_RET_REC_FSID,
     LAYOUTRETURN4_ALL  = LAYOUT4_RET_REC_ALL,
     LAYOUTRETURN4_FILE_NO_ACCESS = LAYOUT4_RET_REC_FILE_NO_ACCESS,
     LAYOUTRETURN4_FSID_NO_ACCESS = LAYOUT4_RET_REC_FSID_NO_ACCESS,
     LAYOUTRETURN4_ALL_NO_ACCESS  = LAYOUT4_RET_REC_ALL_NO_ACCESS
};

struct layoutreturn_file4 {
      offset4         lrf_offset;
      length4         lrf_length;
      stateid4        lrf_stateid;
      /* layouttype4 specific data */
      opaque          lrf_body<>;
};

struct layoutreturn_device_no_access4 {
      deviceid4     lrdna_deviceid;
      nfsstat4      lrdna_status;
};

struct layoutreturn_file_no_access4 {
      offset4         lrfna_offset;
      length4         lrfna_length;
      stateid4        lrfna_stateid;
      deviceid4       lrfna_deviceid;
      nfsstat4        lrfna_status;
      /* layouttype4 specific data */
      opaque          lrfna_body<>;
};

union layoutreturn4 switch(layoutreturn_type4 lr_returntype) {
      case LAYOUTRETURN4_FILE:
              layoutreturn_file4             lr_layout;
      case LAYOUTRETURN4_FILE_NO_ACCESS:
              layoutreturn_file_no_access4   lr_layout_na;
      case LAYOUTRETURN4_FSID_NO_ACCESS:
      case LAYOUTRETURN4_ALL_NO_ACCESS:
              layoutreturn_device_no_access4      lr_device<>;
      default:
              void;
};
```

3.2. RESULT (18.44.2)

The RESULT of the LAYOUTRETURN operation is unchanged; see section 18.44.2 of [RFC5661].

3.3. DESCRIPTION (18.44.3)

The following text is added to the end of the LAYOUTRETURN operation DESCRIPTION in section 18.44.3 of [RFC5661]:

There are three NO_ACCESS layoutreturn_type4 values that indicate a persistent lack of client ability to access storage device(s), LAYOUT4_RET_REC_FILE_NO_ACCESS, LAYOUT4_RET_REC_FSID_NO_ACCESS and LAYOUT4_RET_REC_ALL_NO_ACCESS. A client uses these return types to return a layout (or portion thereof) for a file, return all layouts for an FSID or all layouts from that server held by the client, and in all cases to inform the server that the reason for the return is the client's inability to access one or more storage devices. The same stateid may be used or the client MAY force use of a new stateid in order to report a new error.

An NFS error value (nfsstat4) is included for each device for these three NO_ACCESS return types to provide additional information on the cause. The allowed NFS errors are those that are valid for an NFS READ or WRITE operation, and NFS4ERR_NXIO is also allowed to report an inaccessible device. The server SHOULD log the received NFS error value, but that error value does not affect server processing of the LAYOUTRETURN operation. All uses of the NO_ACCESS layout return types that report NFS errors SHOULD be logged by the client.

The client MAY use the new LAYOUT4_RET_REC_FILE_NO_ACCESS when only one file, or a small number of files are affected. If the access problem affects multiple devices, the client may use multiple file layout return operations; each return operation SHOULD return a layout extent obtained from the device for which an error is being reported. In contrast, both LAYOUT4_RET_REC_FSID_NO_ACCESS and LAYOUT4_RET_REC_ALL_NO_ACCESS include an array of <device, status> pairs to enable a single operation to report errors for multiple devices in a single operation.

3.4. IMPLEMENTATION (18.44.4)

The following text is added to the end of the LAYOUTRETURN operation IMPLEMENTATION in section 18.4.4 of [RFC5661]:

A client that expects to use pNFS for a mounted filesystem SHOULD check for pNFS support at mount time. This check SHOULD be performed by sending a GETDEVICELIST operation, followed by layout-type-specific checks for accessibility of each storage device returned by GETDEVICELIST. If the NFS server does not support pNFS, the GETDEVICELIST operation will be rejected with an NFS4ERR_NOTSUPP error; in this situation it is up to the client to determine whether it is acceptable to proceed with NFS-only access.

Clients are expected to tolerate transient storage device errors, and hence clients SHOULD NOT use the NO_ACCESS layout return types for device access problems that may be transient. The methods by which a client decides whether an access problem is transient vs. persistent are implementation-specific, but may include retrying I/Os to a data server under appropriate conditions.

When an I/O fails because a storage device is inaccessible, the client SHOULD retry the failed I/O via the MDS. In this situation, before retrying the I/O, the client SHOULD return the layout, or inaccessible portion thereof, and SHOULD indicate which storage device or devices was or were inaccessible. If the client does not do this, the MDS may issue a layout recall callback in order to perform the retried I/O.

Backwards compatibility may require a client to perform two layout return operations to deal with servers that don't implement the NO_ACCESS layoutreturn_type4 values and hence respond to them with NFS4ERR_INVAL. In this situation, the client SHOULD perform an ordinary layout return operation and remember that the new layout NO_ACCESS return types are not to be used with that server.

The metadata server (MDS) SHOULD NOT use storage devices in pNFS layouts that are not accessible to the MDS. At a minimum, the server SHOULD check its own storage device accessibility before exporting a filesystem that supports pNFS and when the device configuration for such an exported filesystem is changed (e.g., to add a storage device).

If an MDS is aware that a storage device is inaccessible to a client, the MDS SHOULD NOT include that storage device in any pNFS layouts sent to that client. An MDS SHOULD react to a client return of inaccessible layouts by not using the inaccessible storage devices in layouts for that client, but the MDS is not required to indefinitely retain per-client storage device inaccessibility information. An MDS is also not required to automatically reinstate use of a previously inaccessible storage device; administrative intervention may be required instead.

A client MAY perform I/O via the MDS even when the client holds a layout that covers the I/O; servers MUST support this client behavior, and MAY recall layouts as needed to complete I/Os.

3.4.1. Storage Device Error Mapping (18.44.4.1, new)

The following text is added as new subsection 18.44.4.1 of [RFC5661]:

An NFS error value is sent for each device that the client reports as inaccessible via a NO_ACCESS layout return type. In general:

o If the client is unable to access the storage device, NFS4ERR_NXIO SHOULD be used.

o If the client is able to access the storage device, but permission is denied, NFS4ERR_ACCESS SHOULD be used.

Beyond these two rules, error code usage is layout-type specific:

o For the pNFS file layout, an indicative NFS error from a failed read or write operation on the inaccessible device SHOULD be used.

o For the pNFS block layout, other errors from the Storage Protocol SHOULD be mapped to NFS4ERR_IO. In addition, the client SHOULD log information about the actual storage protocol error (e.g., SCSI status and sense data), but that information is not sent to the pNFS server.

o For the pNFS object layout, occurrences of the object error types specified in [RFC5664] SHOULD be mapped to the following NFS errors for use in LAYOUTRETURN:

```
    o     PNFS_OSD_ERR_EIO            -> NFS4ERR_IO

    o     PNFS_OSD_ERR_NOT_FOUND      -> NFS4ERR_STALE

    o     PNFS_OSD_ERR_NO_SPACE       -> NFS4ERR_NOSPC

    o     PNFS_OSD_ERR_BAD_CRED       -> NFS4ERR_INVAL

    o     PNFS_OSD_ERR_NO_ACCESS      -> NFS4ERR_ACCESS

    o     PNFS_OSD_ERR_UNREACHABLE    -> NFS4ERR_NXIO

    o     PNFS_OSD_ERR_RESOURCE       -> NFS4ERR_SERVERFAULT
```

The LAYOUTRETURN NO_ACCESS return types are used for persistent device errors; they do not replace other error reporting mechanisms that also apply to transient errors (e.g., as specified for the object layout in [RFC5664]).

4. Change to NFS4ERR_NXIO Usage

This document specifies that the NFS4ERR_NXIO error SHOULD be used to report an inaccessible storage device. To enable that usage, this document updates [RFC5661] to allow use of the currently obsolete NFS4ERR_NXIO error in the ARGUMENT of LAYOUTRETURN; NFS4ERR_NXIO remains obsolete for all other uses of NFS errors.

5. Security Considerations

This document adds a small extension to the NFSv4 LAYOUTRETURN operation. The NFS and pNFS security considerations in [RFC5661], [RFC5663] and [RFC5664] apply to the extended LAYOUTRETURN operation.

6. IANA Considerations

There are no additional IANA considerations in this document beyond the IANA Considerations covered in [RFC5661].

7. Conclusions

This draft specifies additions to the pNFS protocol addressing inability to access storage devices used in pNFS layouts.

8. References

8.1. Normative References

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

[RFC5661] Shepler, S., Eisler, M., and D. Noveck, "Network File System (NFS) Version 4 Minor Version 1 Protocol", http://tools.ietf.org/html/rfc5661, January 2010.

[RFC5663] Black, D., Glasgow, J., Fridella, S., "Parallel NFS (pNFS) Block/Volume Layout", http://tools.ietf.org/html/rfc5663, January 2010.

[RFC5664] Halevy, B., Welch, B., Zelenka, J., "Object-Based Parallel NFS (pNFS) Operations", http://tools.ietf.org/html/rfc5664, January 2010

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

Acknowledgments

This draft includes ideas from discussions with the primary author of the pNFS object layout, Benny Halevy, and the Linux kernel pNFS maintainers, including Bruce Fields. In addition, we thank the IETF nfsv4 WG and the following individuals for their comments on prior versions of this draft: Tom Haynes.

This document was prepared using 2-Word-v2.0.template.dot.

Changes from draft-faibish-nfsv4-pnfs-access-permissions-check-03

- First nfsv4 WG draft version.
- Add ALL NO_ACCESS return type, so that there's a NO_ACCESS return type for every current return type.
- Use NFS4ERR_ACCESS instead of NFS4ERR_PERM, and allow use of NFS4ERR_NXIO for an unreachable data server.
- Simplify recommendation for initial access checks to only discuss GETDEVICELIST.
- Add client guidance on riding through transient errors.
- State that server does not need to indefinitely retain device inaccessibility information, and administrative intervention may be required to restore use of a previously inaccessible storage device.
- Remove "MUST" requirement for layout return if retry via MDS fails. Add warning that if the layout isn't returned in advance of MDS I/O retry, the MDS may issue a callback to get it.
- Specify allowed errors (in payload) via reference to errors allowed for READ and WRITE, plus allow NFS4ERR_NXIO.
- Provide information about how to map device errors (especially from non-file layout types) to NFS errors.

Authors' Addresses

Sorin Faibish (editor)
EMC Corporation
228 South Street
Hopkinton, MA 01748
US

Phone: +1 (508) 249-5745
Email: sfaibish@emc.com

David L. Black
EMC Corporation
176 South Street
Hopkinton, MA 01748
US

Phone: +1 (508) 293-7953
Email: david.black@emc.com

Michael Eisler
NetApp
5765 Chase Point Circle
Colorado Springs, CO 80919
US

Phone: +1 (719) 599-9026
Email: mike@eisler.com

Jason Glasgow
Google
5 Cambridge Center, Floors 3-6
Cambridge, MA 02142
US

Phone: +1 (617) 575-1599
Email: jglasgow@google.com