NFSv4 C. Hellwig
Internet-Draft March 23, 2015
Intended status: Standards Track
Expires: September 24, 2015

Parallel NFS (pNFS) SCSI Layout
draft-hellwig-nfsv4-scsi-layout-01.txt

Abstract

Parallel NFS (pNFS) extends Network File Sharing version 4 (RFC5661) to allow clients to directly access file data on the storage used by the NFSv4 server. This ability to bypass the server for data access can increase both performance and parallelism, but requires additional client functionality for data access, some of which is dependent on the class of storage used. The main pNFS operations document specifies storage-class-independent extensions to NFS, the pNFS Block/Volume Layout (RFC5663) specifies the additional extensions for use of pNFS with block-and volume-based storage, while this document provides extensions to the pNFS Block/Volume Layout document to provide reliable fencing and better device discoverability for SCSI based shared storage devices.

Table of Contents

1. Introduction

In the parallel Network File System (pNFS), the metadata server returns Layout Type structures that describe where file data is located. There are different Layout Types for different storage systems and methods of arranging data on storage devices. This document extends the pNFS Block/Volume Layout [RFC5663] with a closer integration into the the SCSI Architecture Model ([SAM-4]) to provide a generic fencing method and more scalable device discovery.

1.1. Scope

This document only specifies an updated version of the layout-specific GETDEVICEINFO XDR response, and a new mandatory fencing method for SCSI devices, but refers to [RFC5663] for the basic principle of operation, as well as the layout specific XDR data structures for the LAYOUTGET and LAYOUTCOMMIT operations. This document does not directly interact with [RFC6688], although the mechanisms described in this document also achieve the goals of [RFC6688], and do so in a more robust fashion that does not depend on the cooperation of the systems involved. Thus, the mechanisms specified in [RFC6688] are not necessary for a pNFS SCSI layout type implementation.

1.2. 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 [RFC2119].

1.3. Code Components Licensing Notice

The external data representation (XDR) description and scripts 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 Section 4 of "Legal Provisions Relating to IETF Documents".

1.4. XDR Description

This document contains the XDR [RFC4506] description of the NFSv4.1 SCSI layout protocol. 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 NFSv4.1 SCSI layout:

```
#!/bin/sh
```

```
grep '^ *///' $* | sed 's?^ *// ??' | sed 's?^ *///$??'
```

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 > flex_files_prot.x
```

The effect of the script is to remove leading white space from each line, plus a sentinel sequence of "///".

The embedded XDR file header follows. Subsequent 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 NFSv4.1 nfs4_prot.x file [RFC5662]. 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.

```
/// /*
///  * This code was derived from draft-hellwig-nfsv4-scsi-layout
///  * Please reproduce this note if possible.
///  */
/// /*
///  * Copyright (c) 2015 IETF Trust and the persons identified
///  * as the document authors.  All rights reserved.
///  *
///  * Redistribution and use in source and binary forms, with
///  * or without modification, are permitted provided that the
///  * following conditions are met:
///  *
///  * - Redistributions of source code must retain the above
///  *   copyright notice, this list of conditions and the
///  *   following disclaimer.
///  *
///  * - Redistributions in binary form must reproduce the above
///  *   copyright notice, this list of conditions and the
///  *   following disclaimer in the documentation and/or other
///  *   materials provided with the distribution.
///  *
///  * - Neither the name of Internet Society, IETF or IETF
///  *   Trust, nor the names of specific contributors, may be
///  *   used to endorse or promote products derived from this
///  *   software without specific prior written permission.
///  *
///  *   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS
///  *   AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED
///  *   WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
```

```
///  *   IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
///  *   FOR A PARTICULAR PURPOSE ARE DISCLAIMED.  IN NO
///  *   EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
///  *   LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
///  *   EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
///  *   NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
///  *   SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
///  *   INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
///  *   LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
///  *   OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
///  *   IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
///  *   ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
///  */
///
/// /*
///  *      nfs4_scsi_layout_prot.x
///  */
///
/// %#include "nfs4_block_layout_prot.x"
/// %#include "nfsv41.h"
///
```

2. SCSI Layout Description

The layout4 type defined in [RFC5662] is extended with a new value as follows:

```
    enum layouttype4 {
        LAYOUT4_NFSV4_1_FILES   = 1,
        LAYOUT4_OSD2_OBJECTS    = 2,
        LAYOUT4_BLOCK_VOLUME    = 3,
        LAYOUT4_SCSI            = 0x80000005
[[RFC Editor: please modify the LAYOUT4_SCSI
  to be the layouttype assigned by IANA]]
    };

    struct layout_content4 {
        layouttype4             loc_type;
        opaque                  loc_body<>;
    };

    struct layout4 {
        offset4                 lo_offset;
        length4                 lo_length;
        layoutiomode4           lo_iomode;
        layout_content4         lo_content;
    };
```

This document defines structure associated with the layouttype4 value LAYOUT4_SCSI. [RFC5661] specifies the loc_body structure as an XDR type "opaque". The opaque layout is uninterpreted by the generic pNFS client layers, but obviously must be interpreted by the Layout Type implementation. All structures behind this opaque value are identical to those defined in [RFC5663].

2.1. GETDEVICEINFO

```
/// /*
///  * Code sets from SPC-3.
///  */
/// enum pnfs_scsi_code_set {
///     PS_CODE_SET_BINARY     = 1,
///     PS_CODE_SET_ASCII      = 2,
///     PS_CODE_SET_UTF8       = 3
/// };
///
/// /*
///  * Designator types from taken from SPC-3.
///  *
///  * Other values are allocated in SPC-3, but not mandatory to
///  * implement or aren't logical unit names.
///  */
/// enum pnfs_scsi_designator_type {
///     PS_DESIGNATOR_EUI64    = 2,
///     PS_DESIGNATOR_NAA      = 3,
///     PS_DESIGNATOR_NAME     = 8
/// };
///
/// /*
///  * Logical unit name + reservation key.
///  */
/// struct pnfs_scsi_base_volume_info4 {
///     pnfs_scsi_code_set             sbv_code_set;
///     pnfs_scsi_designator_type      sbv_designator_type;
///     opaque                         sbv_designator<>;
///     uint32_t                       sbv_pr_key;
/// };
///
```

2.1.1. Model

GETDEVICEINFO calls are handled exactly the same way as specified in [RFC5663]. The "pnfs_scsi_volume_type4" data structure returned by the server as the storage-protocol-specific opaque field da_addr_body in the "device_addr4" structure by a successful GETDEVICEINFO operation [RFC5661] is a strict superset of the

"pnfs_block_volume_type" structured defined by [RFC5663].

2.1.2. Volume Identification

SCSI targets implementing [SPC3] export unique logical unit names for each logical unit through the Device Identification VPD page which can be obtained using the INQUIRY command. This document uses a subset of this information to identify logical units backing pNFS SCSI layouts. It is similar to the "Identification Descriptor Target Descriptor" specified in [SPC3], but limits the allowed values to those that uniquely identify a logical unit. Device Identification VPD page descriptors used to identify logical units for use with pNFS SCSI layouts must adhere to the following restrictions:

1. The "ASSOCIATION" must be set to 0 (The DESIGNATOR field is associated with the addressed logical unit).

2. The "DESIGNATOR TYPE" must be set to one of three values explicitly listed in the "pnfs_scsi_designator_type" enumerations.

The "CODE SET" VPD page field is stored in the "sbv_code_set" field of the "pnfs_scsi_base_volume_info4" structure, the "DESIGNATOR TYPE" is stored in "sbv_designator_type", and the DESIGNATOR is stored in "sbv_designator". Due to the use of a XDR array the "DESIGNATOR LENGTH" field does not need to be set separately. Only certain combinations of "sbv_code_set" and "sbv_designator_type" are valid, please refer to [SPC3] for details, and note that ASCII may be used as the code set for UTF-8 text that contains only ASCII characters. Note that a Device Identification VPD page MAY contain multiple descriptors with the same association, code set and designator type. NFS clients thus MUST iterate the descriptors until a match for "sbv_code_set", "sbv_designator_type" and "sbv_designator" is found, or until the end of VPD page.

Additionally the server returns a Persistent Reservation key in the "sbv_pr_key" field. See Section 2.2 for more details on the use of Persistent Reservations.

2.1.3. Volume Topology

The pNFS SCSI server volume topology is expressed as an arbitrary combination of base volume types enumerated in the following data structures. The individual components of the topology are contained in an array and components may refer to other components by using array indices.

```
/// enum pnfs_scsi_volume_type4 {
///     PNFS_SCSI_VOLUME_SIMPLE =
///          PNFS_BLOCK_VOLUME_SIMPLE ,      /* invalid */
///     PNFS_SCSI_VOLUME_SLICE =             /* see RFC5663 */
///          PNFS_BLOCK_VOLUME_SLICE,
///     PNFS_SCSI_VOLUME_CONCAT =            /* see RFC5663 */
///          PNFS_BLOCK_VOLUME_CONCAT,
///     PNFS_SCSI_VOLUME_STRIPE =            /* see RFC5663 */
///          PNFS_BLOCK_VOLUME_STRIPE,
///     PNFS_SCSI_VOLUME_BASE = 4            /* SCSI LU */
/// };
///


///
/// union pnfs_scsi_volume4 switch (pnfs_scsi_volume_type4 type) {
///     case PNFS_SCSI_VOLUME_SIMPLE:
///         pnfs_block_simple_volume_info4 sv_simple_info;
///     case PNFS_SCSI_VOLUME_SLICE:
///         pnfs_block_slice_volume_info4 sv_slice_info;
///     case PNFS_SCSI_VOLUME_CONCAT:
///         pnfs_block_concat_volume_info4 sv_concat_info;
///     case PNFS_SCSI_VOLUME_STRIPE:
///         pnfs_block_stripe_volume_info4 sv_stripe_info;
///     case PNFS_SCSI_VOLUME_BASE:
///         pnfs_scsi_base_volume_info4 sv_base_info;
/// };
///


/// /* scsi layout specific type for da_addr_body */
/// struct pnfs_scsi_deviceaddr4 {
///     pnfs_scsi_volume4 sda_volumes<>; /* array of volumes */
/// };
///
```

All rules for ordering and formation of a "pnfs_scsi_deviceaddr4" structure are identical to those for a "pnfs_block_deviceaddr4" structure in [RFC5663], except that the new pnfs_scsi_base_volume_info4 PNFS_SCSI_VOLUME_BASE case is used in place of the pnfs_block_simple_volume_info4 PNFS_BLOCK_VOLUME_SIMPLE case as the base structure. A PNFS_BLOCK_VOLUME_SIMPLE element MUST NOT be referenced by a pnfs_scsi_deviceaddr4, but is preserved for XDR level compatibility.

2.2. Client Fencing

[RFC5663] suggests using either LUN masking or cooperative clients to implement client fencing. The first implementation requires the server and the storage device to have a common way to address a client, which is impossible when the NFS and storage connection don't share a network, and requires a non-standardized control protocol between the MDS and the storage device. The second implementation relies on a cooperative client, which is not robust.

Instead this document specifies a new SCSI-specific fencing protocol using Persistent Reservations (PRs), similar to the fencing method used by existing shared disk file systems. By placing a PR of type "Exclusive Access - All Registrants" on each SCSI logical unit exported to pNFS clients the MDS prevents access from any client that does not have an outstanding device device ID that gives the client a reservation key to access the logical unit, and allows the MDS to revoke access to the logic unit at any time.

2.2.1. PRs - Key Generation

To allow fencing individual systems, each system must use a unique Persistent Reservation key. [SPC3] does not specify a way to generate keys. This document assigns the burden to generate unique keys to the MDS, which must generate a key for itself before exporting a volume, and one for each client that accesses a volume. The MDS MAY either generate a key for each client that accesses logic units exported by the MDS, or generate a key for each [logical unit, client] combination. If using a single key per client, the MDS needs to be aware of the per-client fencing granularity.

2.2.2. PRs - MDS Registration and Reservation

Before returning a PNFS_SCSI_VOLUME_BASE volume to the client, the MDS needs to prepare the volume for fencing using PRs. This is done by registering the reservation generated for the MDS with the device using the "PERSISTENT RESERVE OUT" command with a service action of "REGISTER", followed by a "PERSISTENT RESERVE OUT" command, with a service action of "RESERVE" and the type field set to 8h (Exclusive Access - All Registrants). To make sure all I_T nexuses are registered, the MDS SHOULD set the "All Target Ports" (ALL_TG_PT) bit when registering the key, or otherwise ensure the registration is performed for each initiator port.

2.2.3. PRs - Client Registration

Before performing the first IO to a device returned from a GETDEVICEINFO operation the client will register the registration key

returned in sbv_pr_key with the storage device by issuing a "PERSISTENT RESERVE OUT" command with a service action of REGISTER with the "SERVICE ACTION RESERVATION KEY" set to the reservation key returned in sbv_pr_key. To make sure all I_T nexus are registered, the client SHOULD set the "All Target Ports" (ALL_TG_PT) bit when registering the key, or otherwise ensure the registration is performed for each initiator port.

When a client stops using a device earlier returned by GETDEVICEINFO it MUST unregister the earlier registered key by issuing a "PERSISTENT RESERVE OUT" command with a service action of "REGISTER" with the "RESERVATION KEY" set to the earlier registered reservation key.

2.2.4. PRs - Fencing Action

In case of a non-responding client the MDS MUST fence the client by issuing a "PERSISTENT RESERVE OUT" command with the service action set to "PREEMPT" or "PREEMPT AND ABORT", the reservation key field set to the server's reservation key, the service action reservation key field set to the reservation key associated with the non-responding client, and the type field set to 8h (Exclusive Access - All Registrants).

After the MDS preempts a client, all client I/O to the logical unit fails. The client should at this point return any layout that refers to the device ID that points to the logical unit. Note that the client can distinguish I/O errors due to fencing from other errors based on the "RESERVATION CONFLICT" status. Refer to [SPC3] for details.

2.2.5. Client Recovery After a Fence Action

A client that detects I/O errors on the storage devices MUST commit through the MDS, return all outstanding layouts for the device, forget the device ID and unregister the reservation key. Future GETDEVICEINFO calls may refer to the storage device again, in which case a new registration will be performed.

3. Security Considerations

The security considerations in [RFC5663] apply to this document as well.

4. IANA Considerations

IANA is requested to assign a new pNFS layout type in the pNFS Layout Types Registry as follows (the value 5 is suggested): Layout Type Name: LAYOUT4_SCSI Value: 0x00000005 RFC: RFCTBD10 How: L (new layout type) Minor Versions: 1

5. 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", 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.

[RFC5662] Shepler, S., Ed., Eisler, M., Ed., and D. Noveck, Ed., "Network File System (NFS) Version 4 Minor Version 1 External Data Representation Standard (XDR) Description", RFC 5662, January 2010.

[RFC5663] Black, D., Ed., Fridella, S., Ed., and J. Glasgow, Ed., "Parallel NFS (pNFS) Block/Volume Layout", RFC 5663, January 2010.

[RFC6688] Black, D., Ed., Glasgow, J., and S. Faibish, "Parallel NFS (pNFS) Block Disk Protection", RFC 6688, July 2012.

[SAM-4] INCITS Technical Committee T10, "SCSI Architecture Model - 4 (SAM-4)", ANSI INCITS 447-2008, ISO/IEC 14776-414, 2008.

[SPC3] INCITS Technical Committee T10, "SCSI Primary Commands-3", ANSI INCITS 408-2005, ISO/IEC 14776-453, 2005.

Appendix A. Acknowledgments

David Black, Robert Elliott and Tom Haynes provided a throughout review of early drafts of this document, and their input lead to the current form of the document.

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

Christoph Hellwig

Email: hch@lst.de