

Network File System Version 4  
Internet-Draft  
Intended status: Informational  
Expires: December 25, 2019

S. Faibish  
D. Black  
P. Shilane  
Dell EMC  
June 27, 2019

Support for Data Reduction Extended Attributes in nfsv4 Version 2  
draft-faibish-data-reduction-xattrs-pvt-00

## Abstract

This document proposes extending NFSv4 operations to enable file extended attributes or xattr to be used in the protocol to provide information about the data reduction properties of files. New xattrs are proposed to allow the client application to communicate to the NFSv4 server data reduction attributes associated with files and directories using opaque metadata, not interpreted by the file system, but communicated to the Block Storage data reduction engines. Corresponding new file attributes are proposed to allow clients and client applications to query the server for data reduction xattr support and allow to get and set data reduction xattrs on files and directories. Such data reduction metadata is used as hints to the file server about what type of data reduction to apply. The proposed data reduction attributes include achievable ratios for compression and deduplication plus whether each data reduction technique applies to the file.

## Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of Internet-Draft Shadow Directories can be accessed at <https://www.ietf.org/standards/ids/internet-draft-mirror-sites/>.

This Internet-Draft will expire on December 27, 2019.

## Copyright Notice

Internet-Draft      Data Reduction Extended Attributes in NFSv4      June 2019

This document is subject to [BCP 78](https://trustee.ietf.org/license-info) and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

<a href="#">1.</a>	Introduction . . . . .	<a href="#">2</a>
<a href="#">1.1</a>	Terminology . . . . .	<a href="#">3</a>
<a href="#">2.</a>	Use Cases . . . . .	<a href="#">4</a>
<a href="#">3.</a>	Extended Attributes . . . . .	<a href="#">6</a>
<a href="#">4.</a>	File System Support . . . . .	<a href="#">9</a>
<a href="#">5.</a>	Namespaces . . . . .	<a href="#">9</a>
<a href="#">6.</a>	Differences with Named Attributes . . . . .	<a href="#">9</a>
<a href="#">7.</a>	Protocol Enhancements . . . . .	<a href="#">10</a>
<a href="#">8.</a>	IANA Considerations . . . . .	<a href="#">10</a>
<a href="#">9.</a>	Security Considerations . . . . .	<a href="#">10</a>
<a href="#">10.</a>	References . . . . .	<a href="#">11</a>
<a href="#">10.1</a>	Normative References . . . . .	<a href="#">11</a>
<a href="#">10.2</a>	Informative References . . . . .	<a href="#">11</a>
	Acknowledgments . . . . .	<a href="#">12</a>
	Author's Address . . . . .	<a href="#">12</a>

## [1.](#) Introduction

Many NFS servers use expensive solid state media, e.g., NVMe SSDs, complemented by data reduction processing of files to reduce their size on the Block Storage via compression and deduplication, thereby optimizing media usage. This draft considers scenarios in which data reduction processing is performed in Block Storage for NFS servers, i.e., compression and deduplication processing occurs in the background or inline as a consequence of NFS files being written to the Block Storage. In these scenarios, the data reduction engines in Block Storage have limited information about how

reducible (compressible and/or deduplicate-able) the data written to NFS is.

There is additional strong interest to improve data reduction when using NVMe accessed media and exposing such data attributes to the Block Storage as xattrs over NFS is one means of providing this critical to Block Storage data reduction engines.

There is an expired draft for use of NVMe (over fabric) in accessing a pNFS SCSI Layout [3] which could be extended to communicate data reduction attributes to NVMe storage. The shortcoming of the current pNFS SCSI NVMe layout is that it has no information related to data reduction attributes. This document discusses potential use of NFSv4 extended attributes as currently standardized in [2], for communicating additional data reduction metadata; a future version of this document will propose updates to the NFSv4 protocol to support this functionality.

The purpose of this draft is to add xattrs that will allow applications to send richer metadata information to the NFS server in order to optimize Block Storage data reduction engine operations and improve data reduction for data stored by NFS servers.

Applications can handle files with different compression and deduplication characteristics and send this information to the data reduction engines. Current applications have defined data reduction characteristics and there are clear definitions for the typical compression and deduplication ratios of some types of data independent of the application that generated the data. For example electronic data analysis (EDA) has no single de facto standard file extension but generates application files with common compression and deduplication characteristics. Knowing that a file is compressed improves the latency and/or throughput of the NFS server by not attempting to further compress the files. An additional example is that NFS backup of files that are already stored on the Block Storage is likely to result in a very high deduplication ratio.

## [1.1](#) Terminology

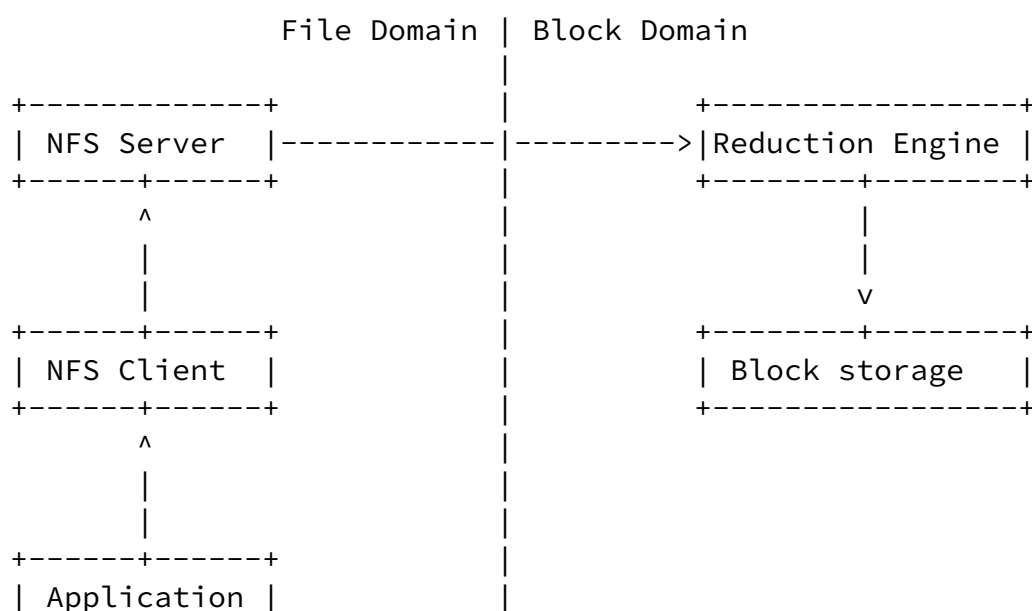
The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this

document are to be interpreted as described in [RFC 2119](#) [1].

In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying [RFC-2119](#) significance. We will refer to the block devices used by the NFS servers as "Block Storage".

## [2.](#) Use Cases

Applications can use extended attributes to store metadata together with the files and directories. Metadata regarding data reduction attributes may be available from applications that use different types of files. This metadata may not be directly useful to the file system but is relevant to the compression and deduplication engines used by the Block Storage to improve data reduction. Use of data reduction metadata is not expected to significantly impact I/O latency or throughput (IOPS).



+-----+

|

Figure 1: Data Reduction Domains for NFSv4

Figure 1 shows the NFSv4 server configuration, data flow and functionality domains with the data reduction engine in the Block domain and located above the Block Storage. This figure represents NFSv4 without parallel NFS (pNFS) support. In this structure the NFS server can communicate xattrs as metadata directly to the Reduction Engine via an extension to the interface to Block Storage.

In general applications using block devices rely on SCSI protocols to access the data. Although SCSI protocols have a rich API, most communication between hosts and Block Storage, e.g., storage arrays, is in terms of blocks, not files. In contrast, applications use large files to read and write data to and from NFS servers. In general, NFS servers use NFS file systems that are stored on SCSI (or NVMe) devices provisioned from Block Storage, e.g., external storage arrays, as Block Storage but file metadata, e.g., file type and file size, is not transferred to the block array in an explicit manner.

An NFS Server might be able to infer data reduction characteristics based on the file type, e.g., a ".mp4" file can be expected to be an MP4 file that contains MPEG-4 content [7]. This is not sufficient due to file content variability, e.g., as a large variety of codecs are used to create MPEG-4 content whose compressibility may vary by codec. To go beyond the file type, the NFS Server could read the file contents to determine compressibility, but this is problematic due to complexity, e.g., the NFS Server may need to parse a significant amount of an MP4 file to obtain the information necessary to understand its compressibility characteristics. This may be impractical if the file is not written to the NFS Server sequentially, and moreover introduces an undesirable dependency on not only the MP4 file format, but also the set of supported codecs that it supports and individual codec characteristics. It is much better to have the application provide information on compressibility, as the application that generates an MP4 file has the information on the file's contents. A mechanism is needed to pass that information to the NFS Server; this document proposes using xattrs.

So, although the xattrs are stored with the files, the current xattr specification [6] indicates that the file system does not understand the structure or content of these extended attributes. If the NFS server could extract the data reduction xattrs and pass their contents to the Block Storage functionality, the Block Storage reduction engines could parse that content and adapt its data reduction behavior accordingly.

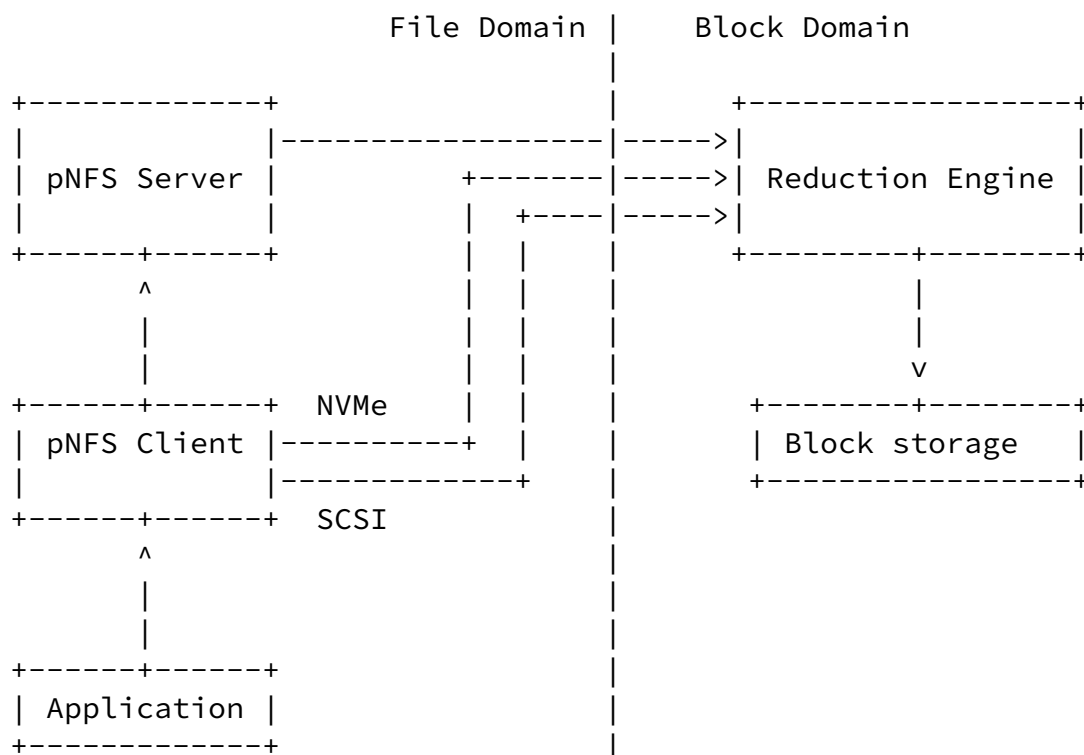


Figure 2: Data Reduction Domains for pNFS over NVMe or SCSI

The current situation is that data reduction done in the block domain lacks critical information that could be provided by the applications in order to improve efficiency of data compression and deduplication.

Figure 2 shows another scenario with a pNFS Server and a block pNFS Client that accesses Block storage using either NVMe or SCSI over a network. In this scenario the pNFS Client could send data reduction attributes directly to the reduction engine above the Block storage layer if the block storage protocol (NVMe or SCSI in the figure) supports doing so. The assumption is that the application has additional information related to files types and typical compression and deduplication parameters associated to

different file types, e.g., see the above discussion of MPEG-4 content. The application can convey this information to the reduction engine to improve the reduction engine efficiency. If the application does not do so, then the user can also add data reduction characteristics for individual files towards improving data reduction efficiency without needing to change the storage array configuration.

For this pNFS scenario the application enables sending data reduction parameters to the Block Device using extensions to the SCSI or NVMe protocols. The pNFS Client still needs to pass the data reduction xattrs to the pNFS Server because the pNFS Client is always allowed to fall back from a pNFS write to an NFS write via the NFS Server; this fallback is similar to the previous case where the NFS Server stores the data reduction xattrs associated with each file.

For example a video application knows whether a file consists of compressed data or uncompressed data. The application writing the data to the pNFS client can set a file attribute that will indicate that a file is uncompressed and hence it is likely to be productive for the data reduction engine to reduce the file's size. The pNFS client passes that information via an xattr that hints that the file is compressible. The pNFS server will change the data reduction xattr and will transmit the xattr to the Block Storage as a hint that the data is uncompressed. The pNFS client will stream the video using the pNFS NVMe data protocol and the compression engine in the Block Storage will compress the data blocks as long as the uncompressed hint is set in NVMe writes from the pNFS Client. If the xattr is changed to indicate that the data has been compressed, the compression engine does not compress the incoming blocks.

A second example is related to encrypted files that can be neither compressed nor deduplicated in the absence of file copying. For this specific example we envision a not-deduplicatable hint.

In this scenario the NFS client sets the deduplication hint to advise to the data reduction engine that deduplication should be enabled for the file. Alternatively if a new file is being written that is not based on modifying an existing file the deduplication hint is set to indicate that deduplication should be disabled.

Another use case involves compressed video files and images that are written by video applications. As such files are already compressed, further attempts to compress them are likely to be pointless, and may negatively impact the performance of the NFS Server.

An additional scenario involves metadata at the start (header) of the file; an application that did not generate the file may nonetheless be able access the metadata section in the file and set extended file attributes based on compression and deduplication found in the file header. The NFS server doesn't have visibility into metadata included in file headers and cannot send file header content to the data reduction engine as separate metadata. Only the user application can access and parse the header and add xattr when the file is written to the NFS server.

Additional examples of known data reduction attributes is implemented in benchmarks such as SPECsfs that is using predefined data reduction attributes. SPECsfs workloads [8] have DR/CR (Deduplication Ratio/Compression Ratio) characteristics that were collected from actual user data. They are as follows:

EDA	DR/CR=50%/50%
SWBUILD	DR/CR=0/80%
VDI	DR/CR=55%/70%
DB	DR/CR=0/50%
VDA	DR/CR=0/0
IT infrastucture	DR/CR=30%/50%
Oracle DW	DR/CR=15%/70%
Oracle OLTP	DR/CR=0%/65%
Exchange 2010	DR/CR=15%/35%
Geoseismic	DR/CR=3%/40%

Another scenario involves placing files with the same known data reduction characteristics in same directory, where the user or an application sets data reduction xattrs the attributes on the directory that are intended to apply to all files in the directory and possibly also sub-directories. In this case the NFS Server uses the data reduction xattrs on the directory to inform the data reduction engine of the data reduction characteristics of blocks in all files in that directory.



### [3](#) Extended Attributes

Extended attributes, also called xattrs [\[6\]](#), are a means to associate opaque metadata with file system objects, e.g., files and directories. Extended attributes are especially useful when they add information that is not, or cannot be, present in the associated object itself. User-space applications can arbitrarily create, read from, and write these attributes.

As extended attributes are file system-agnostic applications do not need to be concerned about how the attributes are stored internally on the underlying file system. All major operating systems provide various flavors of extended attributes. Many user space tools allow xattrs to be included in attributes that need to be preserved when files and directories are updated, moved or copied.

The proposed data reduction attributes are opaque to the file system but can be used by the data reduction engines in the Block Storage reduction engine to increase the data reduction and server operations by viewing the xattrs as hints from the client application regarding file compression and deduplication characteristics. The Block Storage will parse these attributes and change the data reduction methods according to these hints with no need for the file system to know about the data reduction methods used.

Extended attributes have long been considered unsuitable for portability because they are inadequately defined and not formally documented by any standard (such as POSIX). However, evidence suggests that xattrs are widely deployed and their support in modern disk-based file systems is fairly universal. What is different in the new usecase is that the opaque metadata can be received and understood by the data reduction engines. The extended attributes can be 0 or 100 where 0 means "don't do this" hint and 100 is a "do this, but can't predict how much reduction will actually result" hint. They can also take on a percentage value, e.g., from the SPECsfs data shown above.

Any regular file or directory may have a set of extended attributes, each consisting of a key and associated value [\[6\]](#). As currently specified, the NFS client or server MUST NOT interpret the contents of the key or value. This document proposed to remove that restriction in support of data reduction xattrs.

The data reduction attributes can be provided by the extended attributes supported by most modern file systems and can be retrieved from the local file systems on the client and added to the NFS extended attributes when files are moved from local file system attributes of the files to the xattrs in NFS.

## [4](#) File System Support

In Linux, ext3, ext4, JFS, XFS, Btrfs, among other file systems support extended attributes. The `getfattr` and `setfattr` utilities can be used to retrieve and set xattrs. The names of the extended attributes must be prefixed by the name of the category and a dot; hence these categories are generally qualified as name spaces. In the NTFS file system, extended attributes are one of several supported "file streams" [\[5\]](#).

Xattrs can be retrieved and set through system calls, [\[6\]](#), or shell commands and generally supported by user-space tools that preserve other file attributes. For example, the "rsync" remote copy program will correctly preserve extended attributes between Linux/ext4 and OSX/hfs by stripping off the Linux-specific "user." prefix.

## [5](#) Namespaces

Operating systems may define multiple "namespaces" in which xattrs can be set. Namespaces are more than organizational classes; the operating system may enforce different data reduction policies and allow different reduction characteristics depending on the namespace.

## [6](#) Differences with Named Attributes

[RFC5661](#) defines named attributes as opaque byte streams that are associated with a directory or file and referred to by a string name [\[4\]](#). Named attributes are intended to be used by client applications as a method to associate application-specific data with a regular file or directory. In that sense, xattrs are similar in concept and use to named attributes, but there are subtle differences. Named attributes are only visible to the NFS layer and not to the application while extended attributes are accessible to the application layer and can be modified by users. File systems typically define individual xattr "get" and "set" operations. Xattrs generally have size limits ranging from a few bytes to several kilobytes; the maximum supported size is not universally defined and is usually restricted by the file system.

There are no clear indications on how xattrs can be mapped to any existing recommended or optional file attributes defined in [RFC 5661](#) [\[2\]](#); as a result, most NFS client implementations ignore

application-specified xattrs. This results in data loss if one copies, over the NFS protocol, a file with data reduction related xattrs from one file system to another that also supports xattrs. Although different data reduction engines achieve different levels of reduction these attributes are used by the reduction engines to increase the reduction to different levels for different algorithms.

---

Internet-Draft      Data Reduction Extended Attributes in NFSv4      June 2019

While it should be possible to write guidance about how a client can use the named attribute mechanism to act like xattrs, such as carving out some namespace and specifying locking primitives to enforce atomicity constraints on individual get/set operations, this is problematic for data reduction attributes that are specific to specific applications and file types and not defined by the user. As such there will be mechanisms that will detect the reduction attributes from the application or from local file system xattrs.

The different implementations of the protocol would have to address these attributes based on additional guidance such as reserving named some portion of named attribute namespace for xattr-like functionality.

## [7](#) Protocol Enhancements

This section proposes extensions to the NFSv4 protocol operations to allow data reduction xattrs to be queried and modified by clients. A new attribute is added to bitmap4 data type to allow xattr support to be queried. This follows the guidelines specified in [\[2\]](#) with respect to minor versioning. We propose to add 2 bits that will be passed to the reduction engine and used to activate/deactivate the compression and/or the deduplication operations. All the current NFSv4 xattr operations are not changed but we will add 4 new operations, namely GETDRATTR, SETDRATTR, LISTXATTR and REMOVEXATTR to be queried and set. The protocol details will be provided in the next version of the draft.

## [8](#). IANA Considerations

All IANA considerations are covered in [\[4\]](#).

## [9](#). Security Considerations

The additions to the NFS protocol for supporting extended attributes do not alter the security considerations of the NFSv4.1 protocol [4].

Data reduction hints may enable attacks on Block Storage resources that support the NFS Server. Hinting at more data reduction than is possible may cause excessive data reduction processing, and hinting at less data reduction than is possible, including hinting not to perform any data reduction, may result in consumption of more potentially expensive storage capacity. A future version of this draft will discuss what to do about these possible resource attacks.

## [10.](#) References

### [10.1.](#) Normative References

- [1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [2] 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.
- [4] Shepler, S., Ed., Eisler, M., Ed., and D. Noveck, Ed., "Network File System (NFS) Version 4 Minor Version 1 Protocol", [RFC 5661](#), January 2010.

### [10.2](#) Informative References

- [3] C. Hellwig, "Using the Parallel NFS (pNFS) SCSI Layout with NVMe", June 2017,  
<https://tools.ietf.org/html/draft-hellwig-nfsv4-scsi-layout-... nvme-00>
- [5] <http://www.freedesktop.org/wiki/CommonExtendedAttributes>,  
"Guidelines for extended attributes".
- [6] M. Naik, M. Eshel, "File System Extended Attributes in NFSv4"  
<https://datatracker.ietf.org/doc/rfc8276/>

- [7] ISO/IEC 14496-14 "Information technology - Coding of audio-visual objects - Part 14: MP4 file format"
- [8] SPEC SFS 2014 SP2 User's Guide,  
<http://spec.org/sfs2014/docs/usersguide.pdf>

Faibish

Expires December 27, 2019

[Page 11]

---

Internet-Draft      Data Reduction Extended Attributes in NFSv4      June 2019

#### Acknowledgments

This draft has attempted to capture the latest industry trends of adding data reduction attributes needed to increase efficiency of newest flash NVMe technology for file servers. New protocols were proposed specific for NVMe media and we were inspired by new drafts proposed by Christoph Hellwig.

#### Author's Address

Sorin Faibish  
Dell EMC  
228 South Street  
Hopkinton, MA 01774  
United States of America

Phone: +1 508-249-5745  
Email: [faibish.sorin@dell.com](mailto:faibish.sorin@dell.com)

Philip Shilane  
Dell EMC

228 South Street  
Hopkinton, MA 01774  
United States of America

Phone: +1 908-286-7977  
Email: philip.shilane@dell.com

David Black  
DellEMC  
176 South Street  
Hopkinton, MA 01748  
United States of America

Phone: +1 774-350-9323  
Email: david.black@dell.com