

NFSv4 Working Group
Internet Draft
Intended status: Standards Track
Expires: January 2012

D. Hildebrand
IBM Almaden
T. Myklebust
NetApp
S. Falkner
Oracle
July 7, 2011

Support for posix_fadvise
draft-hildebrand-nfsv4-fadvise-02.txt

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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 current Internet-Drafts can be accessed at
<http://www.ietf.org/ietf/lid-abstracts.txt>

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>

This Internet-Draft will expire on January 7, 2011.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://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 BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Abstract

This document proposes a new FADVISE operation for NFSv4.2 to support the posix_fadvise function. FADVISE will communicate expected application behavior to the server, allowing servers to optimize future I/O requests for a file. The posix_fadvise function is supported in Linux and many other operating systems and is starting to be widely used by applications. In addition, the FADVISE operation can communicate other application directions such as the use of direct I/O.

Table of Contents

1. Introduction.....	3
1.1. Requirements Language.....	4
2. POSIX Requirements.....	4
3. Other Requirements.....	5
4. Operation TBD: FADVISE - Send application access pattern hints to server	6

4.1. ARGUMENTS.....	6
4.2. RESULTS.....	7
4.3. DESCRIPTION.....	7
4.4. IMPLEMENTATION.....	8
5. Security Considerations.....	8
6. IANA Considerations.....	8
7. References.....	9
7.1. Normative References.....	9
7.2. Informative References.....	9
8. Acknowledgments.....	9

1. Introduction

NFS is now used in many data centers as the sole or primary method of data access. Consequently, more types of applications are using NFS than ever before, each with their own requirements and generated workloads. This document puts forth a proposal for the NFSv4.2 protocol to support the `posix_fadvise` function [2], allowing applications to communicate their expected behavior to the server.

The `posix_fadvise` operation allows applications to provide hints to the storage system regarding its expected access pattern, e.g., sequential or random, and data re-use behavior, e.g., data range will be read multiple times and should be cached. These hints allow the file system to understand what optimizations it should implement for a specific access to a file. For example, if a application indicates it will never read the data more than once, then the file system can avoid polluting the data cache and not cache the data.

Another instance where applications provide an indication of their desired I/O behavior is when an application specifies the use of direct I/O. This can be done in Linux and AIX via the `open()` `O_DIRECT` parameter and in Solaris via the `directio()` function. Applications specifying the use of direct I/O are telling the file system that it must not cache file data.

While applications can use the `posix_fadvise` function and direct I/O today, with NFS it will only affect behavior on the client. While this can help the NFS client optimize I/O and caching for a file, it does not allow the NFS server and its exported file system to do likewise. For example, with direct I/O, while the client no longer caches data, the NFS server and its exported file system will continue caching data. By caching data that will not be re-read, the server is polluting its cache and possibly causing useful cached data to be evicted.

One option is to modify the existing READ and WRITE operations with FADVISE hints. In the case of READ, optimizations are related to prefetching. In the case of WRITE, FADVISE hints inform the server whether it should write through its read cache or whether it should use an O_DIRECT-like mechanism in order to do an uncached write. In both cases, we're talking about hints that constitute a client's best estimate for how it will be using the data in the future. While that estimate may indeed change, it is only useful to the server if it is stable for a non-zero period of time, i.e., more than a single READ or WRITE operation.

This document adds a new FADVISE operation to communicate the client file access patterns as specified in posix_fadvise to the NFS server. The NFS server upon receiving a FADVISE operation MAY choose to change how it performs I/O and its caching policies, but is under no obligation to do so.

The XDR description is provided 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 metadata layout:

```
#!/bin/sh
grep "^ *///" | sed 's?^ */// ??' | sed 's?^.*///??'
```

I.e. 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 > md.x
```

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

1.1. Requirements Language

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].

2. POSIX Requirements

This proposal is to create a new NFS operation to support the posix_fadvise function, defined as follows [2],

```
int posix_fadvise(int fd, off_t offset, off_t len, int advice);
```

The `posix_fadvise()` function shall advise the implementation on the expected behavior of the application with respect to the data in the file associated with the open file descriptor, `fd`, starting at `offset` and continuing for `len` bytes. The specified range need not currently exist in the file. If `len` is zero, all data following `offset` is specified. The implementation may use this information to optimize handling of the specified data. The `posix_fadvise()` function shall have no effect on the semantics of other operations on the specified data, although it may affect the performance of other operations.

The advice to be applied to the data is specified by the `advice` parameter and may be one of the following values:

`POSIX_FADV_NORMAL` - Specifies that the application has no advice to give on its behavior with respect to the specified data. It is the default characteristic if no advice is given for an open file.

`POSIX_FADV_SEQUENTIAL` - Specifies that the application expects to access the specified data sequentially from lower offsets to higher offsets.

`POSIX_FADV_RANDOM` - Specifies that the application expects to access the specified data in a random order.

`POSIX_FADV_WILLNEED` - Specifies that the application expects to access the specified data in the near future.

`POSIX_FADV_DONTNEED` - Specifies that the application expects that it will not access the specified data in the near future.

`POSIX_FADV_NOREUSE` - Specifies that the application expects to access the specified data once and then not reuse it thereafter.

Upon successful completion, `posix_fadvise()` shall return zero; otherwise, an error number shall be returned to indicate the error.

3. Other Requirements

Many applications do not use or require POSIX semantics. These applications may benefit from additional hints (and points when they are set) that are not covered by `posix_fadvise`. At this point, these hints and requirements are unclear, but may include per-read and per-write hints as well as two additional hints:

Opportunistic Prefetch - This hint indicates that the stateid holder expects to access the data soon; prefetch if it can be done at a marginal cost. The use case for this hint is unclear, since if the

client knows that it will want to read the data soon, then when would it not want the server to prefetch the data at any cost?

Recently Used - The client has recently accessed the byte range in its own cache. This informs the server that the data in the byte range remains important to the client. When the server reaches resource exhaustion, knowing which data is more important allows the server to make better choices about which data to, for example purge from a cache, or move to secondary storage. It also informs the server which delegations are more important, since if delegations are working correctly, once delegated to a client, a server might never receive another I/O request for the file.

The use case for this is also unclear, as most clients already cache data that they know is important and having this data cached twice may be unnecessary. In fact, substantial performance improvements have been demonstrated by making caches more exclusive between each other [8], not the other way around. Other work showed that even infinite sized secondary caches can be largely ineffective [7], but this of course is subject to the workload.

4. Operation TBD: FADVISE - Application access pattern hints to server

The section introduces a new operation, named FADVISE, which allows NFS clients to communicate application file access pattern hints to the NFS server. A new operation is will allow hints to be sent to the server when applications use posix_fadvise, direct I/O, or at any other point at which the client finds useful.

4.1. ARGUMENTS

```
enum fadvise_type {
    FADVISE_NORMAL      = 0,
    FADVISE_SEQUENTIAL  = 1,
    FADVISE_RANDOM      = 2,
    FADVISE_WILLNEED    = 3,
    FADVISE_DONTNEED    = 4,
    FADVISE_NOREUSE     = 5,
};

struct FADVISE4args {
    /* CURRENT_FH: file */
    stateid4      stateid;
    offset4       offset;
    length4       count;
    bitmap4       hints;
};
```

4.2. RESULTS

```
struct FADVISE4resok {  
    bitmap4          hints_res;  
};  
  
union FADVISE4res switch (nfsstat4 _status) {  
    case NFS4_OK:  
        FADVISE4resok fadvise_resok4;  
    default:  
        void;  
};
```

4.3. DESCRIPTION

The FADVISE operation sends an I/O access pattern hint to the server for the owner of stated for a given byte range specified by offset and count. The byte range need not currently exist in the file, but the hint will apply to the byte range when it does exist. The server MAY ignore the advice.

The following are the possible hints:

- o FADVISE_NORMAL - Specifies that the application has no advice to give on its behavior with respect to the specified data. It is the default characteristic if no advice is given for an open file.
- o FADVISE_SEQUENTIAL - Specifies that the application expects to access the specified data sequentially from lower offsets to higher offsets.
- o FADVISE_RANDOM - Specifies that the application expects to access the specified data in a random order.
- o FADVISE_WILLNEED - Specifies that the application expects to access the specified data in the near future.
- o FADVISE_DONTNEED - Specifies that the application expects that it will not access the specified data in the near future.
- o FADVISE_NOREUSE - Specifies that the application expects to access the specified data once and then not reuse it thereafter.

The server will return success if the operation is properly formed, otherwise the server will return an error. The server MUST NOT

return an error if it does not recognize or does not support the requested advice.

The hints_res returned by the server is primarily for debugging purposes and the client SHOULD NOT use this information to change or modify its file access behavior. This is for several reasons. First, the server is under no obligation to carry out any hints that it describes in the hints_res result. Second, the FADVISE operation is a point in time operation, and the server can only respond based upon information at this point in time. As time progresses, the server may need to change its handling of a given file due to several reasons including, but not limited to, memory pressure, additional FADVISE hints sent by other clients, and heuristically detected file access patterns.

The server MAY return different advice than what the client requested. If it does, then this might be due to one of several conditions, including, but not limited to another client advising of a different I/O access pattern; a different I/O access pattern from another client that the server has heuristically detected; or the server is not able to support the requested I/O access pattern, perhaps due to a temporary resource limitation.

4.4. IMPLEMENTATION

The NFS client may choose to issue and FADVISE operation to the server in several different instances. The most obvious is in direct response to an applications execution of posix_fadvise. Another useful point would be when an application indicates it is using direct I/O. Direct I/O may be specified at file open, in which case a FADVISE may be included in the same compound as the OPEN operation with the FADVISE_NOREUSE flag set. Direct I/O may also be specified separately, in which case a FADVISE operation can be sent to the server separately.

5. Security Considerations

None.

6. IANA Considerations

The fadvise_type should be able to be extended.

7. References

7.1. Normative References

- [1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [2] The IEEE and The Open Group, "IEEE Std 1003.1, 2004 Edition, The Open Group Technical Standard Base Specifications, Issue 6", 2004

7.2. Informative References

- [1] Shepler, S., Callaghan, B., Robinson, D., Thurlow, R., Beame, C., Eisler, M., and D. Noveck, "Network File System (NFS) version 4 Protocol", RFC 3530, April 2003.
- [2] Shepler, S., Eisler, M., and D. Noveck, "Network File System (NFS) Version 4 Minor Version 1 Protocol", RFC 5661, January 2010.
- [3] Shepler, S., Eisler, M., and D. Noveck, "Network File System (NFS) Version 4 Minor Version 1 External Data Representation Standard (XDR) Description", RFC 5662, January 2010.
- [4] Nowicki, B., "NFS: Network File System Protocol specification", RFC 1094, March 1989.
- [5] Callaghan, B., Pawlowski, B., and P. Staubach, "NFS Version 3 Protocol Specification", RFC 1813, June 1995.
- [6] S. VanDeBogart, C. Frost, E. Kohler, "Reducing Seek Overhead with Application-Directed Prefetching", in Proceedings of USENIX Annual Technical Conference, June 2009.
- [7] D. Muntz, P. Honeyman, "Multi-level Caching in Distributed File Systems", in Proceedings of USENIX Annual Technical Conference, 1992.
- [8] T.M. Wong, J. Wilkes, "My cache or yours? Making storage more exclusive", in Proceedings of the USENIX Annual Technical Conference, 2002.

8. Acknowledgments

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

Authors' Addresses

Dean Hildebrand
IBM Almaden
650 Harry Rd
San Jose, CA 95120

Phone: +1 408-927-2013
Email: dhildeb@us.ibm.com

Trond Myklebust
NetApp
3215 Bellflower Ct
Ann Arbor, MI 48103
USA

Phone: +1-734-662-6608
Email: Trond.Myklebust@netapp.com

Sam Falkner
Oracle
500 Eldorado Blvd.
Broomfield, CO 80021

Phone: +1 720-279-4303
Email: sam.falkner@oracle.com

