Internet-Draft
Expires: December 2006

Tom Talpey
Brent Callaghan

Document: draft-ietf-nfsv4-nfsdirect-03

June, 2006

NFS Direct Data Placement

Status of this Memo

Abstract

The RDMA transport for ONC RPC provides direct data placement for NFS data. Direct data placement not only reduces the amount of data that needs to be copied in an NFS call, but allows much of the data movement over the network to be implemented in RDMA hardware. This draft describes the use of direct data placement by means of server-initiated RDMA operations into client-supplied buffers in a Chunk list for implementations of NFS versions 2, 3, and 4 over an RDMA transport.

Table of Contents

1. Introduction

The RDMA Transport for ONC RPC [RPCRDMA] allows an RPC client application to post buffers in a Chunk list for specific arguments and results from an RPC call. The RDMA transport header conveys this list of client buffer addresses to the server where the application can associate them with client data and use RDMA operations to transfer the results directly to and from the posted buffers on the client. The client and server must agree on a consistent mapping of posted buffers to RPC. This document details the mapping for each version of the NFS protocol [RFC1831] [RFC1832] [RFC1094] [RFC1813] [RFC3530] [NFSv4.1].

2. Transfers from NFS Client to NFS Server

The RDMA Read list, in the RDMA transport header, allows an RPC client to marshal RPC call data selectively. Large chunks of data, such as the file data of an NFS WRITE request, may be referenced by an RDMA Read list and be moved efficiently and directly-placed by an RDMA READ operation initiated by the server.

The process of identifying these chunks for the RDMA Read list can be implemented entirely within the RPC layer. It is transparent to the upper-level protocol, such as NFS. For instance, the file data portion of an NFS WRITE request can be selected as an RDMA "chunk" within the XDR marshalling code of RPC based on a size criterion,

independently of the NFS protocol layer. The XDR unmarshalling on the receiving system can identify the correspondence between Read chunks and protocol elements via the XDR position value encoded in the Read chunk entry.

RPC RDMA Read chunks are employed by this NFS mapping to convey specific NFS data to the server in a manner which may be directly placed. The following sections describe this mapping for versions of the NFS protocol.

3. Transfers from NFS Server to NFS Client

The RDMA Write list, in the RDMA transport header, allows the client to post one or more buffers into which the server will RDMA Write designated result chunks directly. If the client sends a null write list, then results from the RPC call will be returned as either an inline reply, as chunks in an RDMA Read list of server-posted buffers, or in a client-posted reply buffer.

Each posted buffer in a Write list is represented as an array of memory segments. This allows the client some flexibility in submitting discontiguous memory segments into which the server will scatter the result. Each segment is described by a triplet consisting of the segment handle or steering tag (STag), segment length, and memory address or offset.

```
struct xdr_rdma_segment {
   uint32 handle;    /* Registered memory handle */
   uint32 length;    /* Length of the chunk in bytes */
   uint64 offset;    /* Chunk virtual address or offset */
};

struct xdr_write_chunk {
   struct xdr_rdma_segment target<>;
};

struct xdr_write_list {
   struct xdr_write_chunk entry;
   struct xdr_write_list  *next;
};
```

The sum of the segment lengths yields the total size of the buffer, which must be large enough to accept the result. If the buffer is too small, the server must return an XDR encode error. The server must return the result data for a posted buffer by progressively filling its segments, perhaps leaving some trailing segments unfilled or partially full if the size of the result is less than the total size of the buffer segments.

The server returns the RDMA Write list to the client with the segment length fields overwritten to indicate the amount of data RDMA Written to each segment. Results returned by direct placement must not be

returned by other methods, e.g. by read chunk list or inline. If no result data at all is returned for the element, the server places no data in the buffer(s), but does return zeroes in the segment length fields corresponding to the result.

The RDMA Write list allows the client to provide multiple result buffers - each buffer must map to a specific result in the reply. The NFS client and server implementations must agree on the mapping of results to buffers for each RPC procedure. The following sections describe this mapping for versions of the NFS protocol.

Through the use of RDMA Write lists in NFS requests, it is not necessary to employ the RDMA Read lists in the NFS replies, as described in the RPC/RDMA protocol. This enables more efficient operation, by avoiding the need for the server to expose buffers for RDMA, and also avoiding "RDMA_DONE" exchanges. Clients may additionally employ RDMA Reply chunks to receive entire messages, as described in [RPCRDMA].

4. NFS Versions 2 and 3 Mapping

A single RDMA Write list entry may be posted by the client to receive either the opaque file data from a READ request or the pathname from a READLINK request. The server will ignore a Write list for any other NFS procedure, as well as any Write list entries beyond the first in the list.

Similarly, a single RDMA Read list entry may be posted by the client

to supply the opaque file data for a WRITE request or the pathname for a SYMLINK request. The server will ignore any Read list for other NFS procedures, as well as additional Read list entries beyond the first in the list.

Because there are no NFS version 2 or 3 requests that transfer bulk data in both directions, it is not necessary to post requests containing both Write and Read lists. Any unneeded Read or Write lists are ignored by the server.

In the case where the outgoing request or expected incoming reply is larger than the maximum size supported on the connection, it is possible for the RPC layer to post the entire message or result in a special "RDMA_NOMSG" message type which is transferred entirely by RDMA. This is implemented in RPC, below NFS and therefore has no effect on the message contents.

Non-RDMA (inline) WRITE transfers may optionally employ the "RDMA_MSGP" padding method described in the RPC/RDMA protocol, if the

appropriate value for the server is known to the client. Padding allows the opaque file data to arrive at the server in an aligned fashion, which may improve server performance.

The NFS version 2 and 3 protocols are frequently limited in practice to requests containing less than or equal to 8 kilobytes and 32 kilobytes of data, respectively. In these cases, it is often practical to support basic operation without employing a configuration exchange as discussed in [RPCRDMA]. The server can post buffers large enough to receive the largest possible incoming message (approximately 12KB/36KB would be vastly sufficient in the above cases), and the client can post buffers large enough to receive replies based on the "rsize" it is using to the server. Because the server will never return data in excess of this size, the client can be assured of the adequacy of its posted buffer sizes.

Flow control is handled dynamically by the RPC RDMA protocol, and write padding is optional and therefore may remain unused.

Alternatively, if the server is administratively configured to values appropriate for all its clients, the same assurance of interoperability within the domain can be made.

The use of a configuration protocol with NFS v2 and v3 is therefore optional. Employing a configuration exchange may allow some advantage to server resource management through accurately sizing buffers, enabling the server to know exactly how many RDMA Reads may be in progress at once on the client connection, and enabling client write padding which may be desirable for certain servers when RDMA Read is impractical.

5. NFS Version 4 Mapping

This specification applies to the first minor version of NFS version 4 (NFSv4.0) and any subsequent minor versions that do not override this mapping.

The Write list will be considered only for the COMPOUND procedure. This procedure returns results from a sequence of operations. Only the opaque file data from an NFS READ operation, and the pathname from a READLINK operation will utilize entries from the Write list.

If there is no Write list, i.e. the list is null, then any READ or READLINK operations in the COMPOUND must return their data inline. The NFSv4.0 client must ensure that any result of its READ and READLINK requests must fit within its receive buffers, or an RDMA transport error may occur.

The first entry in the Write list must be used by the first READ or READLINK in the COMPOUND request. The next Write list entry by the by the next READ or READLINK, and so on. If there are more READ or READLINK operations than Write list entries, then any remaining operations must return their results inline.

If a Write list entry is presented, then the corresponding READ or READLINK must return its data via an RDMA WRITE to the buffer indicated by the Write list entry. If the Write list entry has zero RDMA segments, or if the total size of the segments is zero, then the corresponding READ or READLINK operation must return its result inline.

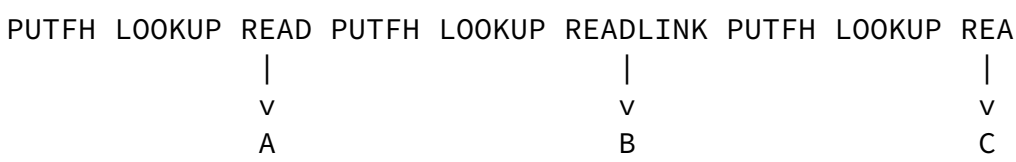
The following example shows an RDMA Write list with three posted buffers A, B, and C. The designated operations in the compound

request, READ and READLINK, consume the posted buffers by writing their results back to each buffer.

RDMA Write list:

A --> B --> C

Compound request:



If the client does not want to have the READLINK result returned directly, then it provides a zero length array of segment triplets for buffer B or sets the values in the segment triplet for buffer B to zeros so that the READLINK result will be returned inline.

The situation is similar for RDMA Read lists sent by the client and applies to the NFSv4.0 WRITE and SYMLINK procedures as for v3. Additionally, inline segments too large to fit in posted buffers may be transferred in special "RDMA_NOMSG" messages.

Non-RDMA (inline) WRITE transfers may optionally employ the "RDMA_MSGP" padding method described in the RPC/RDMA protocol, if the appropriate value for the server is known to the client. Padding allows the opaque file data to arrive at the server in an aligned fashion, which may improve server performance. In order to ensure accurate alignment for all data, it is likely that the client will

restrict its use of optional padding to COMPOUND requests containing only a single WRITE operation.

Unlike NFS versions 2 and 3, the maximum size of an NFS version 4 COMPOUND is unbounded, even when RDMA chunks are in use. While it might appear that a configuration protocol exchange (such as the one described in [RPCRDMA]) would help, in fact the layering issues involved in building COMPOUNDs by NFS make such a mechanism

unworkable. Instead, an extension to NFS version 4 supporting a more comprehensive exchange of upper layer (NFSv4) parameters is proposed in [NFSv4.1]. This proposal also addresses other use of the sizes, such as in the server's response cache.

6. Security

The RDMA transport for ONC RPC supports RPCSEC_GSS security as well as link-level security. The use of RDMA Write to return RPC results does not affect ONC RPC security.

7. IANA Considerations

NFS use of direct data placement may introduce a need for an additional NFS port number assignment for networks which share traditional UDP and TCP port spaces with RDMA services. The iWARP [DDP] [RDMAP] protocol is such an example (Infiniband is not).

NFS servers for versions 2 and 3 [RFC1094] [RFC1813] traditionally listen for clients on UDP and TCP port 2049, and additionally, they register these with the portmapper. NFS servers for version 4 [RFC3050] are required to listen on TCP port 2049, and are not required to register.

An NFS version 2 or version 3 server supporting RPC/RDMA on such a network and registering itself with the RPC portmapper may choose an arbitrary port, or may be assigned an alternative well-known port number for its RPC/RDMA service by IANA. The chosen port must be registered with the RPC portmapper under the netid assigned by the requirement in [RPCRDMA].

An NFS version 4 server supporting RPC/RDMA on such a network must be assigned an alternative well-known port number for its RPC/RDMA service by IANA. Clients will connect to this well-known port without consulting the RPC portmapper (as for NFSv4/TCP).

Any subsequent NFS version 4 minor version's [NFSv4.1] server may reuse port 2049, by requiring the client to perform the RDMA session negotiation supported by this protocol. If it does not require the client to negotiate an RDMA-enabled session, it must use the

alternative port for RPC/RDMA, as for version 4.

This is not an issue on non-IP transports such as native Infiniband, where a non-colliding port translation scheme is used [IBPORT]. On such interfaces, the server can simply listen on the port mapped from the IANA-assigned NFS 2049, or any other port as assigned by the native transport. Such assignments are out of the scope of IANA, and of this document.

8. Acknowledgements

The authors would like to thank Dave Noveck and Chet Juszczak for their contributions to this document.

9. Normative References

[RFC1831]
R. Srinivasan, "RPC: Remote Procedure Call Protocol Specification Version 2",
Standards Track RFC,
http://www.ietf.org/rfc/rfc1831.txt

[RFC1832]
R. Srinivasan, "XDR: External Data Representation Standard",
Standards Track RFC,
http://www.ietf.org/rfc/rfc1832.txt

[RFC1094]
"NFS: Network File System Protocol Specification",
(NFS version 2) Informational RFC,
http://www.ietf.org/rfc/rfc1094.txt

[RFC1813]
B. Callaghan, B. Pawlowski, P. Staubach, "NFS Version 3 Protocol Specification",
Informational RFC,
http://www.ietf.org/rfc/rfc1813.txt

[RFC3530]
S. Shepler, B. Callaghan, D. Robinson, R. Thurlow, C. Beame, M. Eisler, D. Noveck, "NFS version 4 Protocol",
Standards Track RFC,
http://www.ietf.org/rfc/rfc3530.txt

10. Informative References

[RPCRDMA]
T. Talpey, B. Callaghan, "RDMA Transport for ONC RPC"

Internet Draft Work in Progress,
draft-ietf-nfsv4-rpcrdma

[NFSv4.1]
S. Shepler, ed., "NFSv4 Minor Version 1"
Internet Draft Work in Progress,
draft-ietf-nfsv4-minorversion1

[DDP]
H. Shah et al, "Direct Data Placement over Reliable Transports",
Internet Draft Work in Progress,
draft-ietf-rddp-ddp

[RDMAP]
R. Recio et al, "An RDMA Protocol Specification",
Internet Draft Work in Progress,
draft-ietf-rddp-rdmap

[IBPORT]
Infiniband Trade Association, "IP Addressing Annex",
available from www.infinibandta.org

11. Authors' Addresses

Tom Talpey
Network Appliance, Inc.
375 Totten Pond Road
Waltham, MA 02451 USA

Phone: +1 781 768 5329
EMail: thomas.talpey@netapp.com

Brent Callaghan
Apple Computer, Inc.
MS: 302-4K
2 Infinite Loop
Cupertino, CA 95014 USA

EMail: brentc@apple.com

12. Intellectual Property and Copyright Statements

Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any

Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Disclaimer of Validity

Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.