Internet Engineering Task Force (IETF)
Request for Comments: 8167
Category: Standards Track
ISSN: 2070-1721

C. Lever
Oracle

June 2017

Bidirectional Remote Procedure Call on RPC-over-RDMA Transports

Abstract

Minor versions of Network File System (NFS) version 4 newer than minor version 0 work best when Remote Procedure Call (RPC) transports can send RPC transactions in both directions on the same connection. This document describes how RPC transport endpoints capable of Remote Direct Memory Access (RDMA) convey RPCs in both directions on a single connection.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc8167.

Table of Contents

1. Introduction

RPC-over-RDMA transports, introduced in [RFC8166], efficiently convey Remote Procedure Call (RPC) transactions on transport layers capable of Remote Direct Memory Access (RDMA). The purpose of this document is to enable concurrent operation in both directions on a single transport connection using RPC-over-RDMA protocol versions that do not have specific facilities for reverse-direction operation.

Reverse-direction RPC transactions are necessary for the operation of version 4.1 of the Network File System (NFS), and in particular, of Parallel NFS (pNFS) [RFC5661], though any Upper-Layer Protocol (ULP) implementation may make use of them. An Upper-Layer Binding (ULB) for NFS version 4.x callback operation is additionally required (see Section 7) but is not provided in this document.

For example, using the approach described herein, RPC transactions can be conveyed in both directions on the same RPC-over-RDMA version 1 connection without changes to the RPC-over-RDMA version 1 protocol. This document does not update the protocol specified in [RFC8166].

The remainder of this document assumes familiarity with the terminology and concepts contained in [RFC8166], especially Sections 2 and 3.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Understanding RPC Direction

The Open Network Computing Remote Procedure Call (ONC RPC) protocol as described in [RFC5531] is architected as a message-passing protocol between one server and one or more clients. ONC RPC transactions are made up of two types of messages.

A CALL message, or "Call", requests work. A Call is designated by the value CALL in the message's msg_type field. An arbitrary unique value is placed in the message's Transaction ID (XID) field. A host that originates a Call is referred to in this document as a "Requester".

A REPLY message, or "Reply", reports the results of work requested by a Call. A Reply is designated by the value REPLY in the message's msg_type field. The value contained in the message's XID field is copied from the Call whose results are being returned. A host that emits a Reply is referred to as a "Responder".

Typically, a Call results in a corresponding Reply. A Reply is never sent without a corresponding Call.

RPC-over-RDMA is a connection-oriented RPC transport. In all cases, when a connection-oriented transport is used, ONC RPC client endpoints are responsible for initiating transport connections, while ONC RPC service endpoints passively await incoming connection requests.

RPC direction on connectionless RPC transports is not addressed in this document.

2.1. Forward Direction

Traditionally, an ONC RPC client acts as a Requester, while an ONC RPC service acts as a Responder. This form of message passing is referred to as "forward-direction" operation.

2.2. Reverse Direction

The ONC RPC specification [RFC5531] does not forbid passing messages in the other direction. An ONC RPC service endpoint can act as a Requester, in which case, an ONC RPC client endpoint acts as a Responder. This form of message passing is referred to as "reverse-direction" operation.

During reverse-direction operation, the ONC RPC client is responsible for establishing transport connections, even though RPC Call messages come from the ONC RPC server.

ONC RPC clients and servers are optimized to perform and scale well while handling traffic in the forward direction and might not be prepared to handle operation in the reverse direction. Not until NFS version 4.1 [RFC5661] has there been a strong need to handle reverse-direction operation.

2.3. Bidirectional Operation

A pair of connected RPC endpoints may choose to use only forward-direction or only reverse-direction operations on a particular transport connection. Or, these endpoints may send Calls in both directions concurrently on the same transport connection.

"Bidirectional operation" occurs when both transport endpoints act as a Requester and a Responder at the same time.

Bidirectionality is an extension of RPC transport connection sharing. Two RPC endpoints wish to exchange independent RPC messages over a shared connection, but in opposite directions. These messages may or may not be related to the same workloads or RPC Programs.

2.4. XID Values

Section 9 of [RFC5531] introduces the ONC RPC transaction identifier, or "XID" for short. The value of an XID is interpreted in the context of the message's msg_type field.

- The XID of a Call is arbitrary but is unique among outstanding Calls from that Requester.

- The XID of a Reply always matches that of the initiating Call.

When receiving a Reply, a Requester matches the XID value in the Reply with a Call it previously sent.

2.4.1. XID Generation

During bidirectional operation, forward- and reverse-direction XIDs are typically generated on distinct hosts by possibly different algorithms. There is no coordination between forward- and reverse-direction XID generation.

Therefore, a forward-direction Requester MAY use the same XID value at the same time as a reverse-direction Requester on the same transport connection. Though such concurrent requests use the same XID value, they represent distinct ONC RPC transactions.

3. Immediate Uses of Bidirectional RPC-over-RDMA

3.1. NFS Version 4.0 Callback Operation

An NFS version 4.0 client employs a traditional ONC RPC client to send NFS requests to an NFS version 4.0 server's traditional ONC RPC service [RFC7530]. NFS version 4.0 requests flow in the forward direction on a connection established by the client. This connection is referred to as a "forechannel" connection.

An NFS version 4.x "delegation" is simply a promise made by a server that it will notify a client before another client or program running on the server is allowed access to a file. With this guarantee, that client can operate as sole accessor of the file. In particular, it can manage the file's data and metadata caches aggressively.

To administer file delegations, NFS version 4.0 introduces the use of callback operations, or "callbacks", in Section 10.2 of [RFC7530]. An NFS version 4.0 server sets up a forward-direction ONC RPC client, and an NFS version 4.0 client sets up a forward-direction ONC RPC service. Callbacks flow in the forward direction on a connection established between the server's callback client and the client's callback service. This connection is distinct from connections being used as forechannels and is referred to as a "backchannel connection".

When an RDMA transport is used as a forechannel, an NFS version 4.0 client typically provides a TCP-based callback service. The client's SETCLIENTID operation advertises the callback service endpoint with a "tcp" or "tcp6" netid. The server then connects to this service using a TCP socket.

NFS version 4.0 implementations can function without a backchannel in place. In this case, the NFS server does not grant file delegations. This might result in a negative performance effect, but correctness is not affected.

3.2. NFS Version 4.1 Callback Operation

NFS version 4.1 supports file delegation in a similar fashion to NFS version 4.0 and extends the callback mechanism to manage pNFS layouts, as discussed in Section 12 of [RFC5661].

NFS version 4.1 transport connections are initiated by NFS version 4.1 clients. Therefore, NFS version 4.1 servers send callbacks to clients in the reverse direction on connections established by NFS version 4.1 clients.

NFS version 4.1 clients and servers indicate to their peers that a backchannel capability is available on a given transport connection in the arguments and results of the NFS CREATE_SESSION or BIND_CONN_TO_SESSION operations.

NFS version 4.1 clients may establish distinct transport connections for forechannel and backchannel operation, or they may combine forechannel and backchannel operation on one transport connection using bidirectional operation.

Without a reverse-direction RPC-over-RDMA capability, an NFS version 4.1 client additionally connects using a transport with reverse-direction capability to use as a backchannel. Opening an independent TCP socket is the only choice for an NFS version 4.1 backchannel connection in this case.

Implementations often find it more convenient to use a single combined transport (i.e., a transport that is capable of bidirectional operation). This simplifies connection establishment and recovery during network partitions or when one endpoint restarts. This can also enable better scaling by using fewer transport connections to perform the same work.

As with NFS version 4.0, if a backchannel is not in use, an NFS version 4.1 server does not grant delegations. Because NFS version 4.1 relies on callbacks to manage pNFS layout state, pNFS operation is not possible without a backchannel.

4. Flow Control

For an RDMA Send operation to work properly, the receiving peer has to have already posted a Receive buffer in which to accept the incoming message. If a receiver hasn’t posted enough buffers to accommodate each incoming Send operation, the receiving RDMA provider is allowed to terminate the RDMA connection.

RPC-over-RDMA transport protocols provide built-in send flow control to prevent overrunning the number of pre-posted Receive buffers on a connection’s receive endpoint using a "credit grant" mechanism. The use of credits in RPC-over-RDMA version 1 is described in Section 3.3.1 of [RFC8166].

4.1. Reverse-Direction Credits

RPC-over-RDMA credits work the same way in the reverse direction as they do in the forward direction. However, forward-direction credits and reverse-direction credits on the same connection are accounted separately. Direction-independent credit accounting prevents head-of-line blocking in one direction from impacting operation in the other direction.

The forward-direction credit value retains the same meaning whether or not there are reverse-direction resources associated with an RPC-over-RDMA transport connection. This is the number of RPC requests the forward-direction Responder (the ONC RPC server) is prepared to receive concurrently.

The reverse-direction credit value is the number of RPC requests the reverse-direction Responder (the ONC RPC client) is prepared to receive concurrently. The reverse-direction credit value MAY be different than the forward-direction credit value.

During bidirectional operation, each receiver has to decide whether an incoming message contains a credit request (the receiver is acting as a Responder) or a credit grant (the receiver is acting as a requester) and apply the credit value accordingly.

When message direction is not fully determined by context (e.g., suggested by the definition of the RPC-over-RDMA version that is in use) or by an accompanying RPC message payload with a call direction field, it is not possible for the receiver to tell with certainty whether the header credit value is a request or grant. In such cases, the receiver MUST ignore the header's credit value.

4.2. Inline Thresholds

Forward- and reverse-direction operation on the same connection share the same Receive buffers. Therefore, the inline threshold values for the forward direction and the reverse direction are the same. The call inline threshold for the reverse direction is the same as the reply inline threshold for the forward direction, and vice versa. For more information, see Section 3.3.2 of [RFC8166].

4.3. Managing Receive Buffers

An RPC-over-RDMA transport endpoint posts Receive buffers before it can receive and process incoming RPC-over-RDMA messages. If a sender transmits a message for a receiver that has no posted Receive buffer, the RDMA provider is allowed to drop the RDMA connection.

4.3.1. Client Receive Buffers

Typically, an RPC-over-RDMA Requester posts only as many Receive buffers as there are outstanding RPC Calls. Therefore, a client endpoint without reverse-direction support might, at times, have no available Receive buffers.

To receive incoming reverse-direction Calls, an RPC-over-RDMA client endpoint posts enough additional Receive buffers to match its advertised reverse-direction credit value. Each outstanding forward-direction RPC requires an additional Receive buffer above this minimum.

When an RDMA transport connection is lost, all active Receive buffers are flushed and are no longer available to receive incoming messages. When a fresh transport connection is established, a client endpoint posts a Receive buffer to handle the Reply for each retransmitted forward-direction Call, and it posts enough Receive buffers to handle reverse-direction Calls.

4.3.2. Server Receive Buffers

A forward-direction RPC-over-RDMA service endpoint posts as many Receive buffers as it expects incoming forward-direction Calls. That is, it posts no fewer buffers than the number of credits granted in the rdma_credit field of forward-direction RPC replies.

To receive incoming reverse-direction replies, an RPC-over-RDMA server endpoint posts enough additional Receive buffers to handle replies for each reverse-direction Call it sends.

When the existing transport connection is lost, all active Receive buffers are flushed and are no longer available to receive incoming messages. When a fresh transport connection is established, a server endpoint posts a Receive buffer to handle the Reply for each retransmitted reverse-direction Call, and it posts enough Receive buffers to handle incoming forward-direction Calls.

5. Sending and Receiving Operations in the Reverse Direction

The operation of RPC-over-RDMA transports in the forward direction is defined in [RFC5531] and [RFC8166]. In this section, a mechanism for reverse-direction operation on RPC-over-RDMA is defined. Reverse-direction operation used in combination with forward-direction operation enables bidirectional communication on a common RPC-over-RDMA transport connection.

Certain fields in the RPC-over-RDMA header have a fixed position in all versions of RPC-over-RDMA. The normative specification of these fields is contained in Section 4 of [RFC8166].

5.1. Sending a Call in the Reverse Direction

To form a reverse-direction RPC-over-RDMA Call message, an ONC RPC service endpoint constructs an RPC-over-RDMA header containing a fresh RPC XID in the rdma_xid field (see Section 2.4 for full requirements).

The rdma_vers field MUST contain the same value in reverse- and forward-direction Call messages on the same connection.

The number of requested reverse-direction credits is placed in the rdma_credit field (see Section 4).

Whether presented inline or as a separate chunk, the ONC RPC Call header MUST start with the same XID value that is present in the RPC-over-RDMA header, and the RPC header's msg_type field MUST contain the value CALL.

5.2. Sending a Reply in the Reverse Direction

To form a reverse-direction RPC-over-RDMA Reply message, an ONC RPC client endpoint constructs an RPC-over-RDMA header containing a copy of the matching ONC RPC Call's RPC XID in the rdma_xid field (see Section 2.4 for full requirements).

The rdma_vers field MUST contain the same value in a reverse-direction Reply message as in the matching Call message.

The number of granted reverse-direction credits is placed in the rdma_credit field (see Section 4).

Whether presented inline or as a separate chunk, the ONC RPC Reply header MUST start with the same XID value that is present in the RPC-over-RDMA header, and the RPC header's msg_type field MUST contain the value REPLY.

5.3. Using Chunks in Reverse-Direction Operations

A "chunk" refers to a portion of a message's Payload stream that is DDP-eligible and that is placed directly in the receiver's memory by the transport. Chunk data may be moved by an explicit RDMA operation, for example. Chunks are defined in Section 3.4.4 and DDP-eligibility is covered in Section 6.1 of [RFC8166].

Chunks MAY be used in the reverse direction. They operate the same way as in the forward direction.

An implementation might support only ULPs that have no DDP-eligible data items. Such ULPs may use only small messages, or they may have a native mechanism for restricting the size of reverse-direction RPC messages, obviating the need to handle Long Messages in the reverse direction.

When there is no ULP requirement for chunks in the reverse direction, implementers can choose not to provide support for chunks in the reverse direction. This avoids the complexity of adding support for performing RDMA Reads and Writes in the reverse direction.

When chunks are not implemented, RPC messages in the reverse direction are always sent using a Short Message; therefore, they can be no larger than what can be sent inline (that is, without chunks). Sending an inline message larger than the inline threshold can result in loss of connection.

If a reverse-direction requester provides a non-empty chunk list to a Responder that does not support chunks, the Responder MUST reply with an RDMA_ERROR message with rdma_err field set to ERR_CHUNK.

5.4. Reverse-Direction Retransmission

In rare cases, an ONC RPC service cannot complete an RPC transaction and then send a reply. This can be because the transport connection was lost, because the Call or Reply message was dropped, or because the ULP delayed or dropped the ONC RPC request. Typically, the Requester sends the RPC transaction again, reusing the same RPC XID. This is known as an "RPC retransmission".

In the forward direction, the Requester is the ONC RPC client. The client is always responsible for establishing a transport connection before sending again.

With reverse-direction operation, the Requester is the ONC RPC server. Because an ONC RPC server does not establish transport connections with clients, it cannot retransmit if there is no transport connection. It is forced to wait for the ONC RPC client to re-establish a transport connection before it can retransmit ONC RPC transactions in the reverse direction.

If the ONC RPC client peer has no work to do, it can be some time before it re-establishes a transport connection. A waiting reverse-direction ONC RPC Call may time out to avoid waiting indefinitely for a connection to be established.

Therefore, forward-direction Requesters SHOULD maintain a transport connection as long as there is the possibility that the connection peer can send reverse-direction requests. For example, while an NFS version 4.1 client has open delegated files or active pNFS layouts, it maintains one or more transport connections to enable the NFS server to perform callback operations.

6. In the Absence of Support for Reverse-Direction Operation

An RPC-over-RDMA transport endpoint might not support reverse-direction operation (and thus it does not support bidirectional operation). There might be no mechanism in the transport implementation to do so. Or in an implementation that can support operation in the reverse direction, the ULP might not yet have configured or enabled the transport to handle reverse-direction traffic.

If an endpoint is not prepared to receive an incoming reverse-direction message, loss of the RDMA connection might result. Thus, denial of service could result if a sender continues to send reverse-direction messages after every transport reconnect to an endpoint that is not prepared to receive them.

When dealing with the possibility that the remote peer has no transport-level support for reverse-direction operation, the ULP becomes responsible for informing peers when reverse-direction operation is supported. Otherwise, even a simple reverse-direction RPC NULL procedure from a peer could result in a lost connection.

Therefore, a ULP MUST NOT perform reverse-direction ONC RPC operations until the peer has indicated it is prepared to handle them. A description of ULP mechanisms used for this indication is outside the scope of this document.

For example, an NFS version 4.1 server does not send backchannel messages to an NFS version 4.1 client before the NFS version 4.1 client has sent a CREATE_SESSION or a BIND_CONN_TO_SESSION operation. As long as an NFS version 4.1 client has prepared appropriate resources to receive reverse-direction operations before sending one of these NFS operations, denial of service is avoided.

7. Considerations for ULBs

A ULP that operates on RPC-over-RDMA transports may have procedures that include DDP-eligible data items. DDP-eligibility is specified in an Upper-Layer Binding (ULB). Direction of operation does not obviate the need for DDP-eligibility statements.

Reverse-direction-only operation requires the client endpoint to establish a fresh connection. The ULB can specify appropriate RPC binding parameters for such connections.

Bidirectional operation occurs on an already-established connection. Specification of RPC binding parameters is usually not necessary in this case.

For bidirectional operation, other considerations may apply when distinct RPC Programs share an RPC-over-RDMA transport connection concurrently. Consult Section 6 of [RFC8166] for details about what else may be contained in a ULB.

8. Security Considerations

RPC security is handled in the RPC layer, which is above the transport layer where RPC-over-RDMA operates.

Reverse-direction operations make use of an authentication mechanism and credentials that are independent of forward-direction operation but otherwise operate in the same fashion as outlined in Section 8.2 of [RFC8166].

9. IANA Considerations

This document does not require any IANA actions.

10. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC5531] Thurlow, R., "RPC: Remote Procedure Call Protocol Specification Version 2", RFC 5531, DOI 10.17487/RFC5531, May 2009, <http://www.rfc-editor.org/info/rfc5531>.

[RFC5661] Shepler, S., Ed., Eisler, M., Ed., and D. Noveck, Ed., "Network File System (NFS) Version 4 Minor Version 1 Protocol", RFC 5661, DOI 10.17487/RFC5661, January 2010, <http://www.rfc-editor.org/info/rfc5661>.

[RFC7530] Haynes, T., Ed. and D. Noveck, Ed., "Network File System (NFS) Version 4 Protocol", RFC 7530, DOI 10.17487/RFC7530, March 2015, <http://www.rfc-editor.org/info/rfc7530>.

[RFC8166] Lever, C., Ed., Simpson, W., and T. Talpey, "Remote Direct Memory Access Transport for Remote Procedure Call Version 1", RFC 8166, DOI 10.17487/RFC8166, June 2017, <http://www.rfc-editor.org/info/rfc8166>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <http://www.rfc-editor.org/info/rfc8174>.

Acknowledgements

Tom Talpey was an indispensable resource, in addition to creating the foundation upon which this work is based. The author's warmest regards go to him for his help and support.

Dave Noveck provided excellent review, constructive suggestions, and navigational guidance throughout the process of drafting this document.

Dai Ngo was a solid partner and collaborator. Together we constructed and tested independent prototypes of the changes described in this document.

The author wishes to thank Bill Baker and Greg Marsden for their unwavering support of this work. In addition, the author gratefully acknowledges the expert contributions of Karen Deitke, Chunli Zhang, Mahesh Siddheshwar, Steve Wise, and Tom Tucker.

Special thanks go to Transport Area Director Spencer Dawkins, NFSV4 Working Group Chair and Document Shepherd Spencer Shepler, and NFSV4 Working Group Secretary Tom Haynes for their support.

Author's Address

Charles Lever
Oracle Corporation
1015 Granger Avenue
Ann Arbor, MI 48104
United States of America

Phone: +1 248 816 6463
Email: chuck.lever@oracle.com