RPC-over-RDMA
Version Two

Chuck Lever
<chuck.lever@oracle.com>
Outline

• Relevant documents and their status
• Overview of new features
• Open issues
• Next steps
This Working Group document defines a new version of the RPC-over-RDMA transport protocol.

Replaced draft-cel-nfsv4-rpcrdma-nfsv4-version-two in November 2019, now at revision -01.

• This Working Group document defines bindings between the NFS family of protocols and RPC-over-RDMA version 2.

• New document as of November 2019, now at revision -01.

• WG milestone: Submit final document December 2020.
Protocol Integration

• Previous slides didn’t mention certain ancestor documents:
  • Reverse-direction operation (RFC 8167) is now specified as part of RPC-over-RDMA version 2.
  • Capability probing (rpcrdma-cm-pvt-data) is now handled in RPC-over-RDMA version 2, instead of being exchanged via Communication Manager Private Data.
Feature Overview
Performance

• NFSv4 OPEN, GETATTR, LOOKUP from a Linux client all require Reply chunks. RPC/RDMA version 2 reduces the need for explicit RDMA operations for small and moderately-sized RPC messages by introducing:
  
  • Larger default inline thresholds
  
  • Message continuation
  
  • Extra context switches needed on client to invalidate memory. RPC/RDMA version 2 integrates support for remote invalidation
Extensibility

- Together these facilities enable one-way messages, control plane messages, and other extensions that can be defined later

  - XDR definition changes
  - Feature probing
  - Flow control improvements
Reply Size Estimation

- When Reply does not fit in provisioned Write/Reply chunks:
  - New error codes enable specific Requester recovery actions.
  - Message continuation can often be used instead of a Reply chunk.
NFS ULB version 2

• Reply size estimation requirements have been relaxed considerably:

  • When a Requester provisions an inadequate or no Reply chunk, the Responder can use Message Continuation.

  • When a Responder returns an error reporting the provisions it needs to send the Reply, the Requester can retry with correctly-sized RDMA Reply resources.
Security

- Peer authentication
  - Relies on both property exchange and message continuation.
A Closer Look
XDR Extensibility

- RPC/RDMA version 1

```c
// enum rdma_proc {
//    RDMA_MSG = 0,
//    RDMA_NOMSG = 1,
//    RDMA_MSGP = 2,
//    RDMA_DONE = 3,
//    RDMA_ERROR = 4
// };

// union rdma_body switch (rdma_proc proc) {
//    case RDMA_MSG:
//      rpc_rdma_header rdma_msg;
//    case RDMA_NOMSG:
//      rpc_rdma_header_nomsg rdma_nomsg;
//    case RDMA_MSGP:
//      rpc_rdma_header_padded rdma_msgp;
//    case RDMA_DONE:
//      void;
//    case RDMA_ERROR:
//      rpc_rdma_error rdma_error;
// };

// struct rdma_msg {
//    uint32    rdma_xid;
//    uint32    rdma_vers;
//    uint32    rdma_credit;
//    rdma_body rdma_body;
// };
```

- RPC/RDMA version 2

```c
// struct rpcrdma_common {
//    uint32         rdma_xid;
//    uint32         rdma_vers;
//    uint32         rdma_credit;
//    uint32         rdma_htype;
// };

// struct rpcrdma2_hdr_prefix {
//    struct rpcrdma_common       rdma_start;
//    uint32                      rdma_flags;
// };

// struct rpcrdma2_chunk_lists {
//    uint32                      rdma_inv_handle;
//    struct rpcrdma2_read_list   *rdma_reads;
//    struct rpcrdma2_write_list  *rdma_writes;
//    struct rpcrdma2_write_chunk *rdma_reply;
// };

// const rpcrdma2_proc RDMA2_MSG = 0;
// const rpcrdma2_proc RDMA2_NOMSG = 1;
// const rpcrdma2_proc RDMA2_ERROR = 4;
// const rpcrdma2_proc RDMA2_CONNPROP = 5;

// struct rpcrdma2_msg {
//    struct rpcrdma2_chunk_lists rdma_chunks;
//    uint32                      rdma_rpc_first_word;
// };

// struct rpcrdma2_nomsg {
//    struct rpcrdma2_chunk_lists rdma_chunks;
// };
```
## Transport Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Code</th>
<th>XDR type</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max Send size</td>
<td>1</td>
<td>uint32</td>
<td>4096</td>
</tr>
<tr>
<td>Receive Buffer size</td>
<td>2</td>
<td>uint32</td>
<td>4096</td>
</tr>
<tr>
<td>Max segment size</td>
<td>3</td>
<td>uint32</td>
<td>1048576</td>
</tr>
<tr>
<td>Max segment count</td>
<td>4</td>
<td>uint32</td>
<td>16</td>
</tr>
<tr>
<td>Reverse-direction support</td>
<td>5</td>
<td>uint32</td>
<td>0</td>
</tr>
<tr>
<td>Host Authentication Token</td>
<td>6</td>
<td>opaque&lt;&gt;</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Credits & Flow Control

- Enable RPC-over-RDMA to support asymmetrical operation: a message in one direction might trigger zero, one, or multiple messages in the other direction in response.

- Credits are requested and granted in both directions: 32-bit rdma_start.rdma_credit field is split into a pair of 16-bit subfields

- An asynchronous credit grant mechanism was added: RDMA2_NOMSG with empty chunk lists
Message Continuation

- Sender sets the RDMA2_F_MORE flag.
- Receiver concatenates the data payload of the next received message to the end of the data payload of the received message. There is no protocol-defined limit on the number of concatenated messages in a sequence.
- Sender clears the RDMA2_F_MORE flag in the final message in the sequence.
- Sender includes chunks only in the final message in a sequence.
- Credit exhaustion can occur at the receiver in the middle of a sequence of continued messages.
Open Issues
Read Chunks

- RPC/RDMA v1 allows a position zero Read chunk to appear in an RDMA_MSG type Call. Where does a Responder put the inline portion of such a message?

- RPC/RDMA v1 does not explicitly require an RDMA_NOMSG type Call to have a position zero Read chunk. Does such a message have gaps? Are they zero-filled?

- RPC/RDMA v1 does not prevent or prohibit overlapping Read chunks. Is the correct response ERR CHUNK?
Remote Invalidation

- Remote invalidation is currently not limited to RDMA2_MSG and RDMA2_NOMSG type messages.

- For instance, should a Responder be permitted to use Send With Invalidate when posting an RDMA2_ERROR type message?

- Or, no constraints here, and allow Responder implementers flexibility?
Next Steps

• Review these documents.

• There are no prototype implementations yet. Prototypes will help identify and resolve ambiguities, controversies, and open issues.

• Milestone states document delivery by December 2020. Is there a plan for WGLC?