RPC/RDMA Credit Accounting
chucklever@gmail.com
In Brief

• Motivation for draft-cel-nfsv4-rpcrdma-version-two

  • RPC/RDMA transport protocols use Credit-Based Flow Control to avoid congestion and connection loss

  • The existing credit accounting mechanism in RPC/RDMA v1 does not support our goals for RPC/RDMA v2

  • Which RPC/RDMA v1 shortcomings shall we address?
RPC/RDMA v1 Credits

- RPC/RDMA flow control is \textit{credit-based}, as opposed to pause, loss, or rate-based
  - Receiver grants “credits” to sender based on the number of buffers or amount of buffer space the receiving endpoint has
  - Sender waits for credits to be granted before transmitting
RPC/RDMA v1 Credits

• One credit equals one RPC transaction (Call + Reply), no matter how much data is transferred, or whether it involves additional RDMA data transfers

• In each RPC Reply, the responder grants credits to requester. The Reply acts as both a credit grant and an ACK for previous activity

• One credit is available upon connection establishment (i.e., before the first Reply is transmitted on that connection)
RPC/RDMA v1 Credits

- Credit grants are delivered *in-band* as part of each message

- *End-to-end* – per Reliable Connection

- *Non-windowing* – The total number of available receive buffers, rather than the number of unconsumed receive buffers, is reported as the grant

- *Adaptive* – The responder’s credit grant can change during the lifetime of the connection
RPC/RDMA v1
Shortcomings

• Does not support cases where no RPC transaction is involved
  • Control plane messages with no RPC XID
  • Connection-level keep-alive
RPC/RDMA v1

Shortcomings

• Does not support unpaired messages
  • Retransmission of RPC Calls
  • RPC Call with no Reply, like unicast or broadcast
  • RDMA_DONE or similar
  • Unsolicited Sends from responder to requester
RPC/RDMA v1
Shortcomings

- Does not support cases where ratio of transport Send to RPC message is not 1:1
  - Multiple RPC messages in one transport Send
  - Multiple exchanges for a single RPC transaction
  - A single RPC message requiring multiple transport Sends
RPC/RDMA v1
Shortcomings

• Bi-directional RPC is problematic

  • Two directions equals two responders, therefore there has to be one credit grant per direction

  • RPC/RDMA v1 re-uses the one "credits" field

• In a single-sided message on a bi-directional connection, what does the "credits" field mean?

  • RPC/RDMA sniffs RPC calldir field; if no RPC message, no calldir field to sniff…
RPC/RDMA v1
Shortcomings

- No mechanism to resynchronize if one side loses track of credits

- Non-windowing credit accounting is inherently resilient to loss of credit grant, but not to loss of a data packet

- Only recourse is to break the connection to re-initialize credit accounting
RPC/RDMA v1
Shortcomings

- Provides no network Quality of Service guarantees
- No way to protect against noisy neighbors or DoS
- Lower bound of one on credit grants. No way to request a larger lower bound
RPC/RDMA v2 Goals

• Incremental performance improvements
  • Larger default inline threshold, remote invalidation

• Extensibility as part of the base protocol
  • Richer error reporting

• Transport property negotiation
  • Ability to send something other than a single RPC message per RDMA Send
Example Extensions

- Transmitting a moderately-sized RPC message using multiple Sends rather than an RDMA data transfer
  - Slide 8

- Requesting cancellation of an ongoing RPC transaction
  - Slides 6 and 8

- Returning an arbitrarily large RPC Reply without overrunning a Reply chunk
  - Slides 6, 7, and 8
Additional Issues

- First tier support for reverse direction operation
  - The use of DDP and Remote Invalidation in the reverse direction
  - Slide 9 (error reporting)
- RPC retransmission
  - Slide 10
Some Possible Fixes

• Gate Sends rather than RPC transactions (no XDR change)

• Change from a non-windowing to a windowing scheme (no XDR change)

• Add a second credits field to the Transport Header. Each message would carry a credit request and a credit grant, and would apply to both directions concurrently

• Add an RDMA_ACK proc that conveys current grants, to act as ACK of an unpaired Send