NFS/RDMA Draft Status

Tom Talpey
Network Appliance
tmt@netapp.com

NFS/RDMA Internet-Drafts

- RDMA Transport for ONC RPC
  - Basic ONC RPC transport definition for RDMA
  - Transparent, or nearly so, for all ONC ULPs
- NFS Direct Data Placement
  - Maps NFS v2, v3 and v4 to RDMA
- NFSv4 RDMA and Session extensions
  - Transport-independent Session model
  - Enables reliability semantics by bounding size of request cache
  - Sharpens v4 over RDMA

ONC RPC over RDMA

- Internet Draft, republished in July
  - draft-ietf-nfsv4-rpcrdma-00
  - Brent Callaghan and Tom Talpey
- Defines new RDMA RPC transport type
- Goal: Performance
  - Achieved through use of RDMA for copy avoidance
  - No semantic extensions

RPC/RDMA Update

- Clarified some issues:
  - Chunking only impacts data, counts remain in non-RDMA part of message!
  - Resolved ambiguity in double lengths – one in data and one in chunk (they must agree)
  - RDMA_DONE consumes a credit
- Published as working group item

NFS Direct Data Placement

- Internet Draft, republished in July
  - draft-ietf-nfsv4-nfsdirect-00
  - Brent Callaghan and Tom Talpey
- Defines NFSv2 and v3 operations mapped to RDMA
  - READ and READLINK
- Also defines NFSv4 COMPOUND
  - READ and READLINK

NFS Direct update

- Minor clarifications:
  - Mentioned WRITE
  - Added to introductory text
- Published as working group item

Others

- NFS/RDMA Problem Statement
  - Republished in July
  - Boilerplate, references etc. only
  - draft-ietf-nfsv4-nfs-rdma-problem-statement-01
- NFS/RDMA Requirements
  - Published December 2003
  - Needs a refresh

NFSv4/Sessions update

V4/Sessions Internet Draft

- Updated and republished in July
  - draft-ietf-nfsv4-sess-00
  - Tom Talpey, Spencer Shepler and Jon Bauman
- Note slightly new name
  - Sessions are the primary motivator
  - RDMA still addressed/enabled
- Significant new implementation section

The Proposal

- Add a session to NFSv4
- Transport-independent
- Enable operation on single connection
  - Firewall-friendly
- Enable multiple connections for trunking, multipathing
- Enable server resource control
- *Enable implementing reliable semantics*

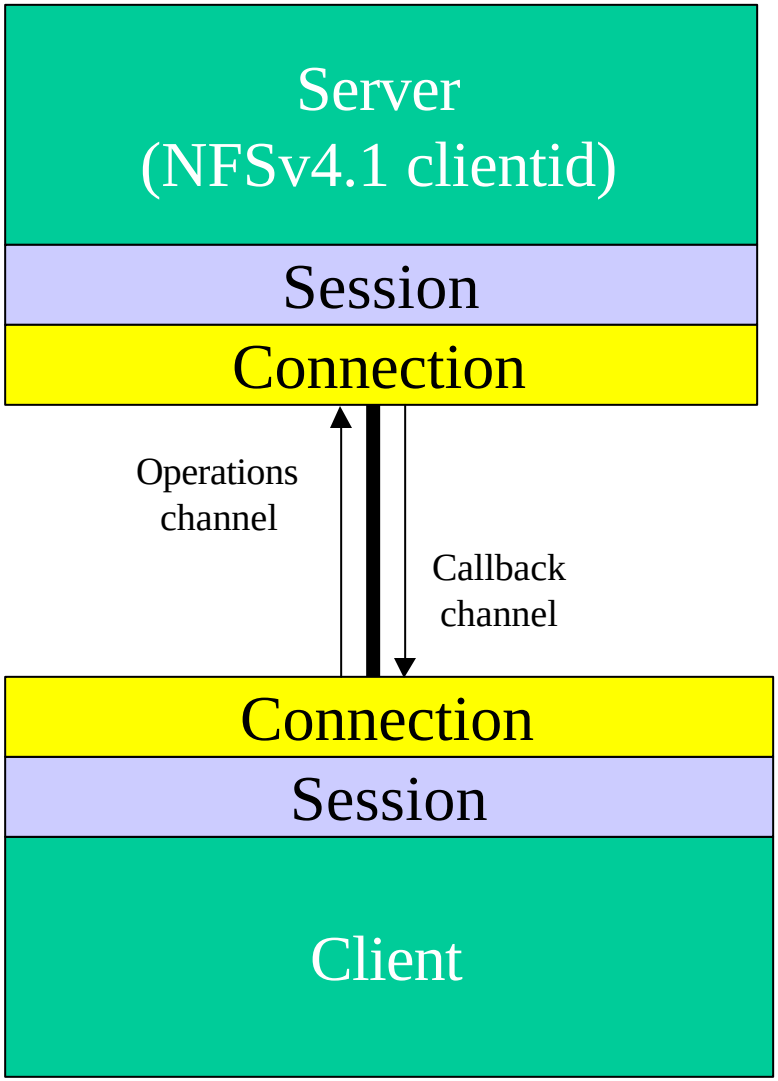
Session

- New object added to protocol
- Sits on top of transport connections
- Client connections “bind” to session
- Session members share a clientid

Session Connection Model

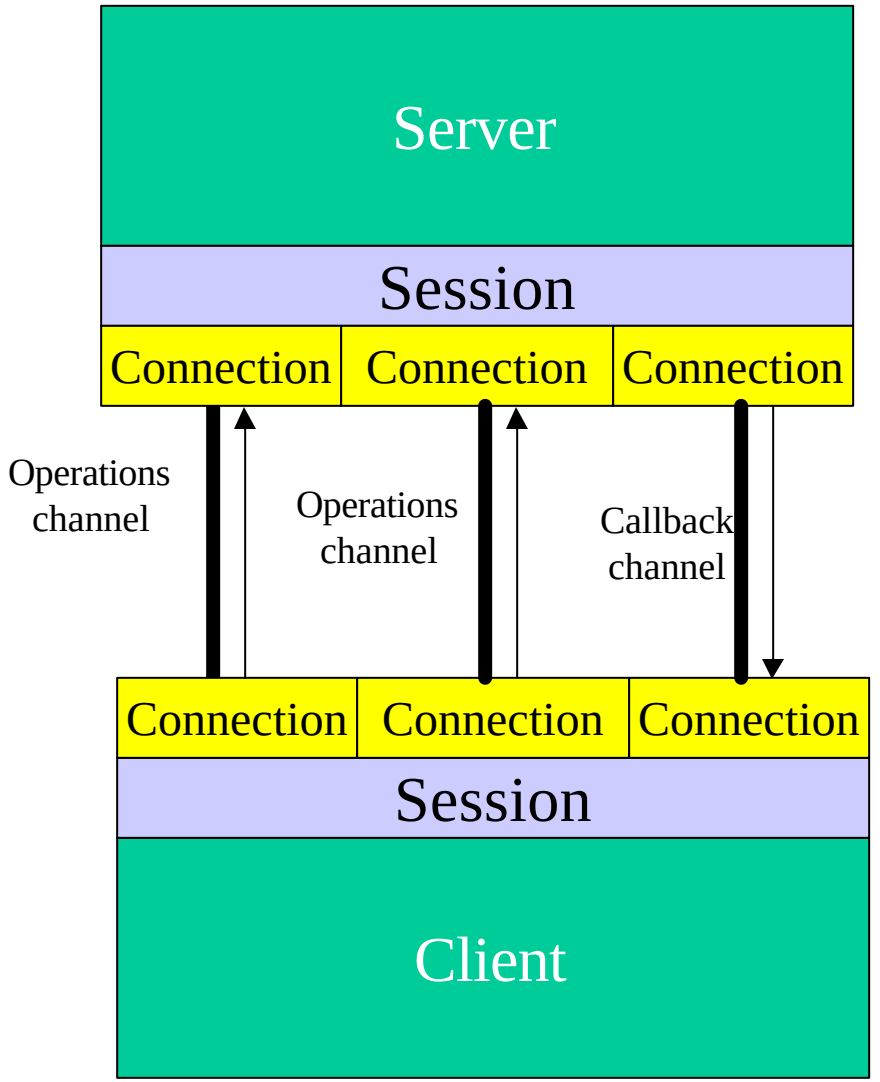
- Client connects to server
- First time only:
  - Create new session and clientid
- Initialize channel:
  - Bind backchannel
  - Operation channel(s) simply assert session
  - May bind operations, callback to same connection
  - May connect additional times
- CCM fits here by reference from session
- If connection lost, “reconnect” to existing session
- When done, destroy session context

Example Session – single connection



- Resource-friendly
- Secure/Firewall-friendly
- No performance impact

Example Session – multiple connections



- Resource-friendly
- Secure
- Trunked/multipathed

Reliability Semantics

- Highly desirable, but never achievable
  => server's request cache would have unbounded size
- Need flow control (N) , operation sizing (M)
  - (in order to support RDMA too)
- Flow control provides an "ack window"
  - Use this to retire request cache entries
- N * M = request cache size
- Session provides accounting and storage
- Request cache implementation enables Exactly-Once

Slotid (formerly streamid)

- A per-operation identifier in the range 0..N-1 of server's current flow control
  - In effect, an index into an array of legal in-progress ops
- Highly efficient processing – no XID lookup
- Used in conjunction with other session info to maintain request cache

XID handling

- XIDs are not unique!
- Only unique in context of single connection and RPC program
  - http://www.cthon.org/talks96/werme1.pdf
- Many issues across reconnect
- SEQUENCE corrects this:
  - {session, slot, sequence} triplet is unique and bounded in the request cache

V4 Protocol integration

- Piggyback on existing COMPOUND
- New session control first in each session COMPOUND request and reply
- Conveys clientid/sessionid, slotid, etc.
- Each request within session renews leases

Updated Proposal

- New concepts/objects:
  - Session/sessionid
  - Slot/Slotid
  - (I.e. no more channelid)
  - Some name changes from previous
- “Session” now subject to clientid
  - As it should be
- Session BIND performed only on backchannel
  - Operations channels merely assert membership
- Chaining removed (as discussed in Ann Arbor)

Operations

- CREATECLIENTID
- CREATESESSION
- BIND_BACKCHANNEL
- DESTROYSESSION
- SEQUENCE
- [CB_SEQUENCE] - todo
- CB_RECALLCREDIT

Session Creation ops

- CREATECLIENTID
  - Because setclientid passes callback addr
  - New operation passes only nfs_client_id4
- CREATESESSION
  - Takes clientid and session parameters
  - Persistent request cache boolean, sizes
  - For now, transport attributes (TBD future)

Backchannel binding

- Only backchannel requires bind
  - Because we still need the “second half” of setclientid – after session creation
  - Takes clientid, callback program, ident, sizes
  - Also (for now) transport attributes

SEQUENCE

- Passes session information in each op
  - Sessionid, sequenceid, slotid, maxslot, chaining
- Sessionid gives context (membership)
- Slotid indexes into request cache
- {session, sequence, slot} triplet is response cache uniqueness token
- Sequenceid provides replay detection

Slotid handling

- Slotid ranges from 0-maxrequests
- Index into server response cache
- Client-chosen per op
- Maxslot is client highwater of in-use slotids
  - Provides server early slot retirement
  - Provides client advance slot hinting
- Sequenceid changes on each use of slotid
  - Server use as replay indicator

Reconnect

- Original proposal required BIND on each new connection or reconnection
- RPC library implementation issues
  - Needed NFSv4 layer op “in between”
- Now, no bind required on operations channel connect
- RPC library can implement transparently

Protocol To-Dos

- OPEN_CONFIRM interaction
- Sequenceid's – really ignored?
- CB_SEQUENCE
- Session creation replay
- Server persistence period and rules
- Transport attributes
- Usage scenarios