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 exactly-once semantics
  - Sharpens v4 over RDMA

ONC RPC over RDMA

- Internet Draft, last published 2003
  - draft-callaghan-rpcrdma-01
  - 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

- Update and republish as WG draft
- Clarify some issues:
  - Chunking only impacts data, counts remain in non-RDMA part of message!
  - Resolve ambiguity in double lengths – one in data and one in chunk (they must agree)
  - RDMA_DONE consumes a credit
- Currently in progress

NFS Direct Data Placement

- Internet Draft, last published 2003
  - draft-callaghan-nfsdirect-01
  - 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

- Update and republish as WG draft
- Minor clarifications:
  - Mention WRITE
  - Add to introductory text
- Currently in progress

Others

- NFS/RDMA Problem Statement
  - Published February 2004
  - draft-ietf-nfsv4-nfs-rdma-problem-statement-00
- NFS/RDMA Requirements
  - Published December 2003

NFSv4/Sessions update

Tom Talpey
Network Appliance
tmt@netapp.com

The Proposal

- Add a session to NFSv4
- Enable operation on single connection
  - Firewall-friendly
- Enable multiple connections for trunking, multipathing
- *Provide Exactly-Once semantics*
- Transport-independent
- Enable resource accounting (sizes, etc)

Session

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

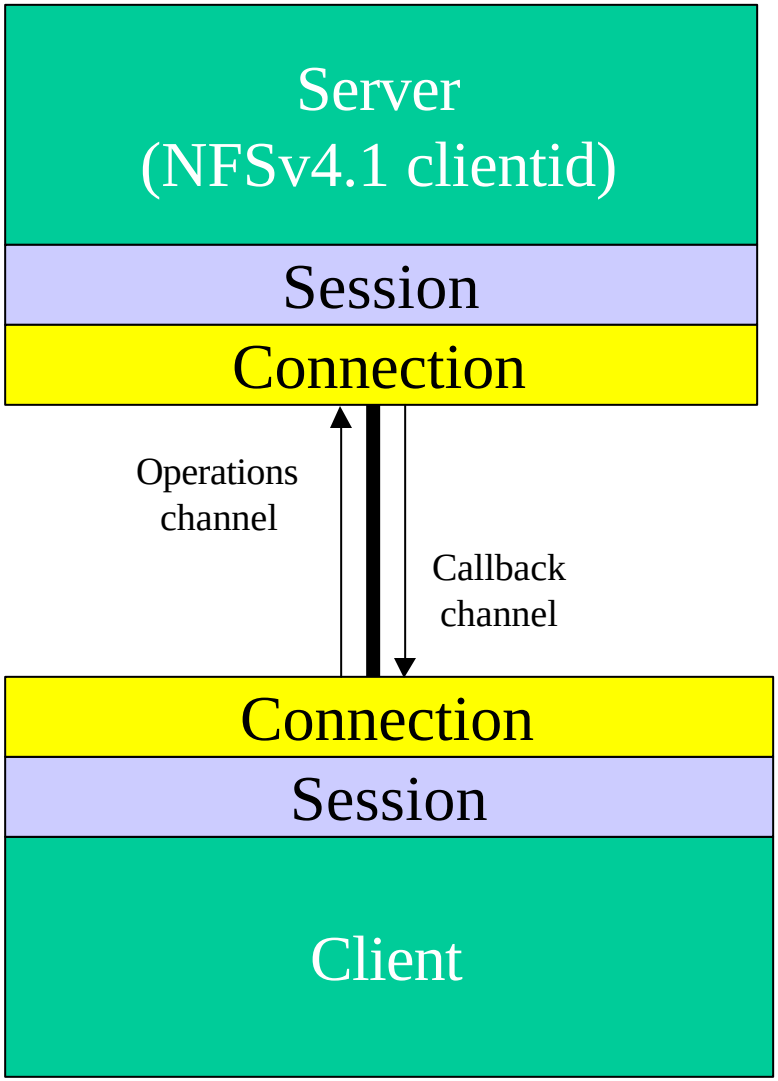
Channels versus Connections

- Channel: a connection bound to a specific purpose:
  - Operations (1 or more connections)
  - Callbacks (typically 1 connection)
- Multiple connections per client, multiple channels per connection
  - Many-to-many relationship
- All operations require a streamid/session
  - Encoded into COMPOUND

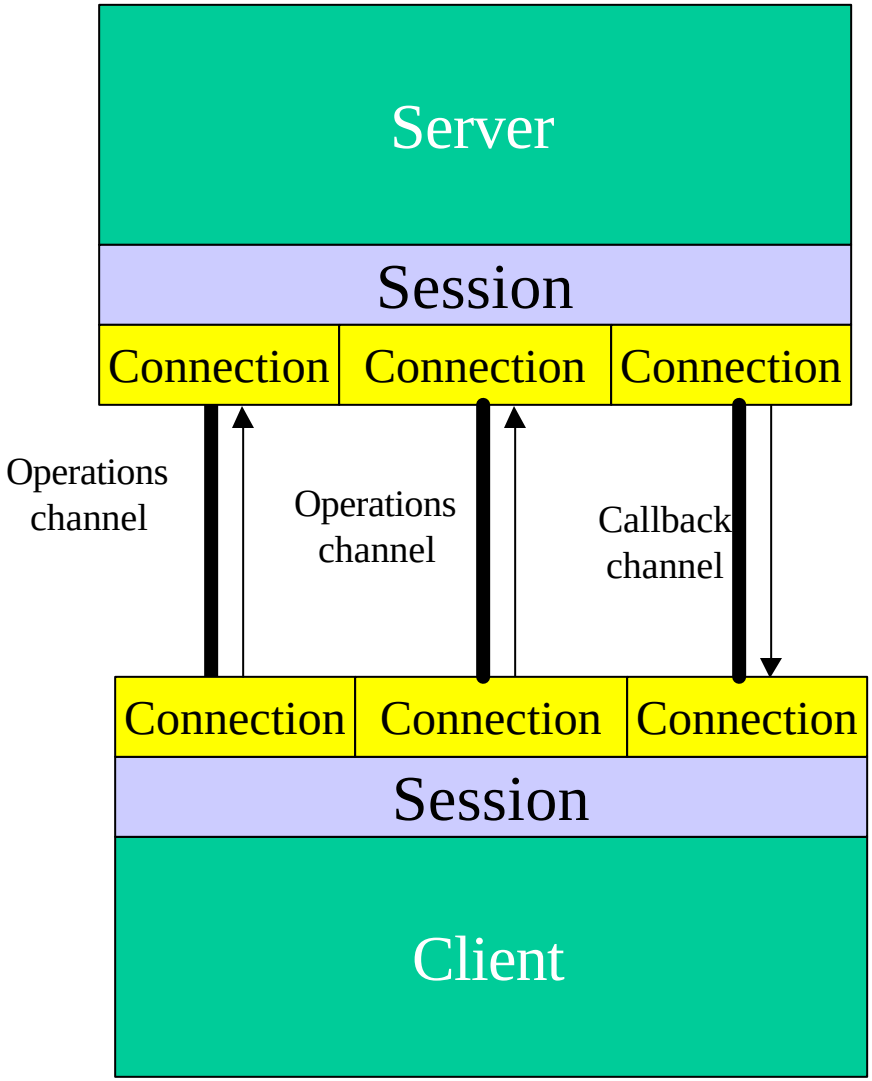
Session Connection Model

- 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
    - Trunking, multipathing, failover, etc.
- CCM fits here by reference from session
- If connection lost, may reconnect to existing session
- When done, destroy session context

Example Session – single connection



Example Session – multiple connections



Example Session – single connection

- Resource-friendly
- Firewall-friendly
- No performance impact
- Isn’t this the way callbacks should have been spec’ed?

Exactly-Once Semantics

- Highly desirable, but never achievable
- Need flow control (N) , operation sizing (M) in order to support RDMA
- Flow control provides an “ack window”
  - Use this to retire response cache entries
- N * M = response cache size
- Session provides accounting and storage
- Done!

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 lookup
- Used in conjunction with other session info to maintain duplicate request cache
  - (not necessarily RPC transaction id)

Chaining

- Problem: COMPOUND restricted in length at session negotiation
- Chaining provides strict sequencing of requests
  - “compound for compounds”
- Start, middle, end flags (and none)
- Maintains current and saved filehandles like COMPOUND

Connection model and negotiation

- Simplest form – no session at all (V4)
- Session create/binding enables use of exactly-once
- Session holds request/response sizes, credits, etc.
- Connection RDMA mode follows transport
  - Per-channel (connection) transport mode
  - Mix TCP and RDMA channels per-client!
- Statically enabled RDMA (e.g. Infiniband) supported
  - Requires preposted buffer

V4 Protocol integration

- Piggyback on existing COMPOUND
- New session control first in each session COMPOUND request and reply
- Conveys session, slotid, and chaining

V4 efficiencies

- No need for sequenceid
  - Field will stay, but ignored under a session
- Each request within session renews leases
- OPEN_CONFIRM not needed
- CCM is enabled

Original Proposal

- Concepts/objects were:
  - Session/sessionid
  - Channel/channelid
  - Stream/Streamid
- “Session” mostly replaced Clientid
- Session BIND was required prior to first op on a new channel

Original Proposal: 5 new ops

- SESSION_CREATE
- SESSION_BIND
- SESSION_DESTROY
- OPERATION_CONTROL
- CB_CREDITRECALL

Updated Proposal

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

Update: 6 new ops

- CREATECLIENTID
- CREATESESSION
- BIND_BACKCHANNEL
- DESTROYSESSION
- SEQUENCE
- 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 reply 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

- Was called “OPERATION_CONTROL”
- Passes session information in each op
  - Sessionid, sequenceid, slotid, maxslot, chaining
- Sessionid gives context (membership)
- Slotid indexes into response cache
- {session, sequence, slot} triplet is response cache uniqueness token

Slotid handling

- Slotid (was streamid) ranges 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

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
  - Response cache can use for exactly-once

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

New nfs_prot_41.x

- Draft of prototype
- See file

Implementation Status

- Prototyping v4/sessions/TCP
- CITI Linux Client
- CITI Linux Server
- NetApp Server
- Operational at basic level today
  - No trunking
  - Anyone want to see a demo?
- RDMA next, when Linux 2.6 kDAPL working

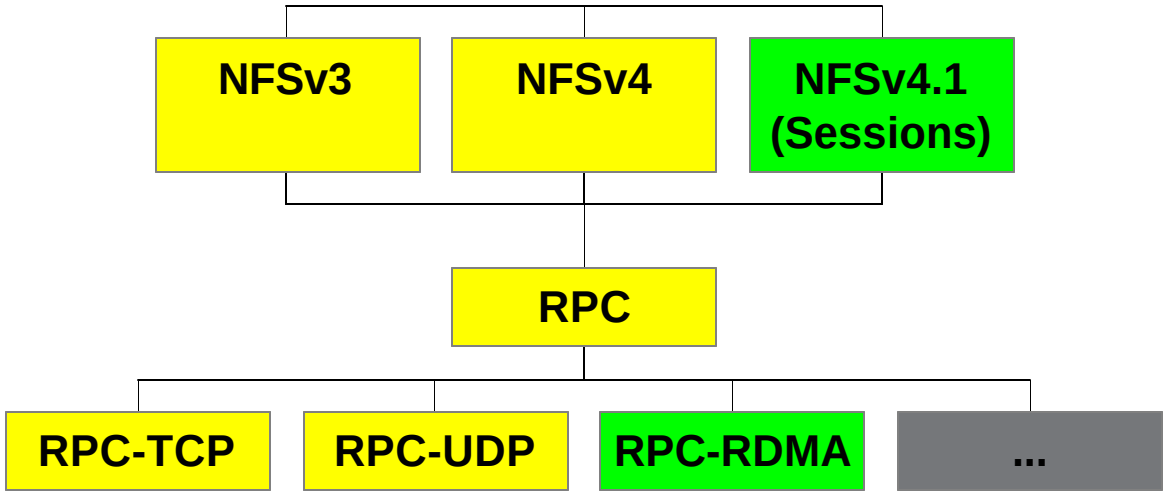
NFS/RDMA Linux Client Update

Tom Talpey

Network Appliance, Inc.

tmt@netapp.com

NFS-RDMA Protocol Stack



Client Implementation Goals

- Support NFSv*/RDMA, NFSv4/Sessions
- Support other transports:
  - IPv6
  - TOE
  - “Bypass” (pNFS)
- Integrate with Linux

Existing Linux RPC support

- Single module – sunrpc.o
- Only IPPROTO_{TCP,UDP}
- Only kernel sockets API
- Much specific knowledge roto-tilled:
  - Stream/dgram (framing needed)
  - Connection oriented (reconnect needed)
  - Reliable (retransmit needed)
- Endpoint is 1-1 per xprt (mount)

Solution: RPC Transport Switch

- Abstraction for transport type
- One each for
  - TCP
  - UDP
  - RDMA
  - More to come
- Server switch TBD

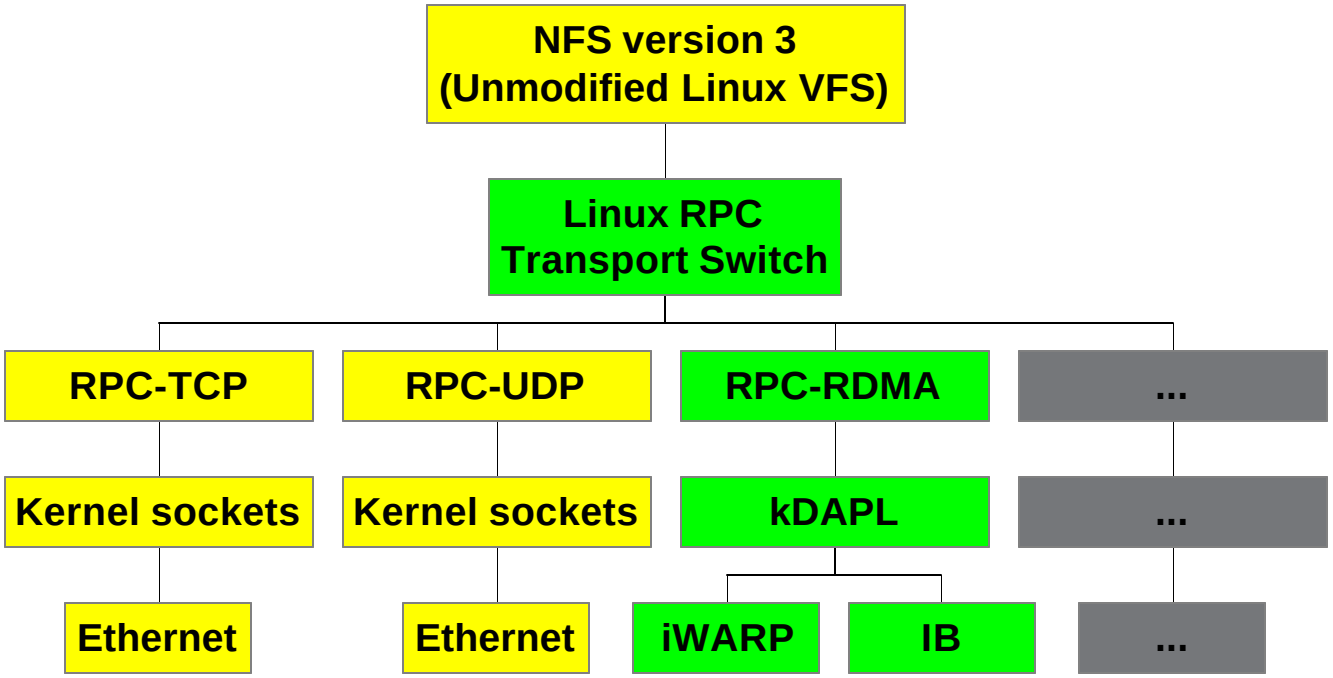
Client Implementation

- Available as open source
  - BSD-style license
  - www.sourceforge.net/projects/nfs-rdma
- Supported Linuxes:
  - RedHat 7.3 (2.4.18)
  - SuSE 8 Enterprise (2.4.19)
  - RHEL 3.0 (2.4.21)
  - 2.6 support under way

2.4 Client Implementation

- Patch for sunrpc (transport switch)
- RPC/RDMA module
  - 3000 lines of code, 2 headers, 3 C files
- kDAPL “null” provider
- IB kDAPL providers under way

NFS-RDMA
Client Software Stack



2.6 Transport Switch Vector

New pointer in the “struct rpc_xprt”:

```
struct xprt_procs {
        void            (*setbufsize)(struct rpc_xprt *);
        void            (*connect)(void *);
        fastcall int     (*send_request)(struct rpc_task *);
        void            (*close)(struct rpc_xprt *);
        void            (*destroy)(struct rpc_xprt *);
};
```

2.6 Transport Switch

- TCP, UDP coded to use new switch
- Significant cleanup from it!
- IPv6 work under way
- TOE is TBD

2.6 implementation patches

- See Chuck Lever if interested
- Targeting 2.6 kernel.org sometime later this year
- RPC/RDMA module nearly ready
- Will appear on Sourceforge
  - www.sourceforge.net/projects/nfs-rdma