RPC-Over-RDMA Version Two

Chuck Lever, Oracle

Focused Changes

- Relieve obvious performance constraints
- Facilitate incremental protocol improvements
- Leverage current Version One implementations
- Continue to support broad set of hardware and user space and embedded applications

Extensibility

- Enable improvements:
  - Over time to address issues and fully develop and prototype new OPTIONAL features
  - Without protocol version bump or specification update
  - Without specification and prototyping dependencies between unrelated features

Extensibility Mechanism

- Base protocol
  - Defines an opcode field and an opaque data field in the RPC-over-RDMA header
- Extensions
  - Define new opcodes and message data types in other standards track documents
  - Concatenate XDR definitions from these documents to build a sparse feature set

Extensibility XDR

```
/* From RFC 5531 Section 9 */
enum msg_type {
    CALL = 0,
    REPLY = 1,
};

struct rpcrdma2_optional {
    enum msg_type rdma_optdir;
    uint32 rdma_opttype;
    opaque rdma_optinfo<>;
};
```

Extensibility XDR

```
enum rpcrdma2_proc {
     RDMA2_MSG = 0,
     RDMA2_NOMSG = 1,
     RDMA2_ERROR = 4,
     RDMA2_OPTIONAL = 5
};
```

```
union rpcrdma2_body switch (rpcrdma2_proc rdma_proc) {
     case RDMA2_MSG:
          rpcrdma2_chunks rdma_chunks;
     case RDMA2_NOMSG:
          rpcrdma2_chunks rdma_chunks;
     case RDMA2_ERROR:
          rpcrdma2_error rdma_error;
     case RDMA2_OPTIONAL:
          rpcrdma2_optional rdma_optional;
};
```

Extensibility XDR

```
enum rpcrdma2_errcode {
    RDMA2_ERR_VERS = 1,
    RDMA2_ERR_BAD_HEADER = 2,
    RDMA2_ERR_INVAL_OPTION = 3
};

union rpcrdma2_error switch (rpcrdma2_errcode rdma_err) {
    case RDMA2_ERR_VERS:
        rpcrdma2_err_vers rdma_vrange;
    case RDMA2_ERR_BAD_HEADER:
        void;
    case RDMA2_ERR_INVAL_OPTION:
        void;
};
```

4KB Inline Threshold

- RDMA Read for small WRITE/SYMLINK payloads adds registration plus a round trip
- Average size of NFSv4 metadata operations is greater than similar NFSv3 operations
  - Large RPCs require a Reply and/or a Position Zero Read chunk (a registration/invalidation)
- Backchannel operations without RDMA are size-constrained

Why Not More Than 4KB?

- The minimum value allowed for Version Two should be something that can be broadly implemented
- 4KB can be supplemented by OPTIONAL features:
  - Exchange of inline threshold maxima
  - Message continuation
- Large Receive buffers pin lots of memory
- There might not be much benefit past 16KB

Remote Invalidation?

- Send With Invalidate is a Send that asks receiving RNIC to invalidate one tag before Receive completes

- Receive completion reports the invalidated tag to the consumer

- Smart implementations might utilize only one tag per RPC, mitigating cost of invalidating during RPC reply completion

Remote Invalidation?

- Basic support requires no XDR changes
  - State that Version Two responders MAY use Send With Invalidate in place of Send to convey replies
    - OPTIONAL features can later refine when a server actually uses Send With Invalidate
- V2 is then unusable on clients with RNICs that do not handle Send With Invalidate
  - Does this disenfranchise interesting hardware or deployment scenarios?