RDMA Bonding

Liran Liss
Mellanox Technologies
Agenda

• Introduction
• Transport-level bonding
• RDMA bonding design
• Recovering from failure
• Implementation
Bonding (Link Aggregation)

- Bond together multiple physical links into a single aggregate logical link

- Motivation
  - Aggregate bandwidth (active-active)
    - Distribute communication flows across all active links
  - High availability (active-backup)
    - If a link goes down, reassign traffic to remaining links

- Can we do the same for HCAs?
Link-level Bonding

- Example: Ethernet link aggregation
- Typically accomplished by a "Bonding" pseudo network interface
- Placed between the L3/4 stack and physical interfaces
  - Multiplexes **packets** across **stateless** network interfaces
  - Transparent to higher levels of the stack
  - Transport is implemented in SW

- RDMA challenge
  - Transport implemented at stateful network interfaces (HCAs)
Session-level Bonding

- Example: iSCSI
  - Initiator establishes a session with Target
    - Session may comprise multiple TCP flows
  - Connections are completely encapsulated within the iSCSI session
    - OS issues SCSI commands
  - Alternatively, multiple sessions may be created to the same target/LUN
    - May be presented as single logical LUN by multi-path SW

- RDMA challenge
  - Transport connections visible to ULPs
  - Multiple RDMA consumers
Idea: Transport-level Bonding

• Provided by a pseudo-HCA (vHCA)
• Applications open virtual resources
  – vPDs, vQPs, vSRQs, vCQs, vMRs
  – Mapped to physical resources by vHCA
• Namespace translated on the fly
  – Similar to transparent RDMA migration
    • IBM/OSU “Nomad” paper
    • VMware vRDMA
    • Oracle live-migration prototype

• Link aggregation
  – Distribute QPs across HCAs
  – Optionally bond different HCA types
  – Upon failover
    • Reconnect over a different device/port
    • Continue traffic from the point of failure
  – Transparent migration is a special HA case
Requirements

- Support aggregate across different physical HCAs
  - Optionally even different device types
- HW independent Bonding driver
- Strict semantics
  - Adhere to transport message ordering guarantees
  - Global visibility of all IO operations
- Transparent to consumers
  - Including failover events
- High performance
Design

- **User-space solution**
  - Bond driver is a Verbs provider
  - Uses RDMACM internally
    - To open connections
    - Negotiate state using private data

- **IP addressing**
  - GID = IP
  - QPN = Port number
  - HCA identity = alias IP

- **1:1 virtual → physical QP mapping**
  - Leverage HW ordering guarantees
  - Zero copy messages

- **Fast path done in app context**
  - Post_Send(), Post_Recv(), PollCQ()
Posting WRs

• If vQP is not in a suitable state or virtual queue is full
  – Return immediate error

• **Enqueue WR in virtual Queue**

• If associated HW Send / Receive queue is full
  – Return with success

• **For Sends**
  – If connection is not active
    • Schedule (re)connection and return with success
  – For UD
    • Resolve AH and remote QPN (if not already cached)
  – For RDMA
    • Resolve RKey (if not already cached)

• **For Receives**
  – If connection is not active, return with success

• **Translate local SGE**

• **Post to HW**
Polling Completions

• Poll next HW CQ associated with vCQ
• If not empty, process according to status
  – Case IBV_WC_RETRY_EXC_ERR
    • Schedule reconnection for associated vQP
    • Ignore completion
  – Case IBV_WC_WR_FLUSH_ERR
    • Ignore completion
  – Case IBV_WC_SUCCESS
    • Report successful completion
  – Default (any other error)
    • Modify vQP to error
    • Report erroneous completion
    • Add corresponding virtual Queue to CQ error list
• Poll next virtual queue on error list
• If it has in-flight WQEs
  – Generate ERROR_FLUSH for next WQE
• Report CQ empty if none of the above applies
RC Failure Recovery

- Re-establish connection
  - Over any active link and device
- Negotiate last committed operations
  - Generate corresponding completions
- Rewind physical queues
  - Resume operation
RC Failure Recovery

• Re-establish connection
  – Over any active link and device
• Negotiate last committed operations
  – Generate corresponding completions
• Rewind physical queues
  – Resume operation

Send Queue

Receive Queue
RC Failure Recovery

- Re-establish connection
  - Over any active link and device
- Negotiate last committed operations
  - Generate corresponding completions
- Rewind physical queues
  - Resume operation
RC Failure Recovery

• Re-establish connection
  – Over any active link and device

• Negotiate last committed operations
  – Generate corresponding completions

• Rewind physical queues
  – Resume operation
RC Failure Recovery

- Re-establish connection
  - Over any active link and device
- Negotiate last committed operations
  - Generate corresponding completions
- Rewind physical queues
  - Resume operation
Implementation (Ongoing)

• Current status
  – POC implementation
  – Supported objects
    • CQs
    • PDs
    • RC QPs
    • MRs
  – Supported operations
    • Resource manipulation
    • Send-receive data traffic
  – QPs limited to single link
    • Tackle transient link failure

• Next steps
  – Complete Verbs coverage
  – RDMACM integration
  – Multi-link recovery
    • Continuously negotiate active links
  – Aggregation schemes
    • HA
    • RR
    • Static load balancing
    • Dynamic load balancing
Summary

• Bonding solution for stateful RDMA devices
  – HW agnostic
  – Aggregates ports from different devices
  – Communicating peers must run the Bonding driver
    • Out-of-band protocol via CM MADs

• Supports
  – High availability
  – Aggregate BW
  – Load balancing
  – Transparent migration

• Efficient user-space implementation
  – Could be extended to the kernel in a similar manner
Thank You