Next Generation Hardware Assists And Scalability

Dror Goldenberg
Mellanox Technologies Inc.

Agenda

- Scaling to Large Clusters
- Reliable Multicast
- Stateless TCP/IP Offloads

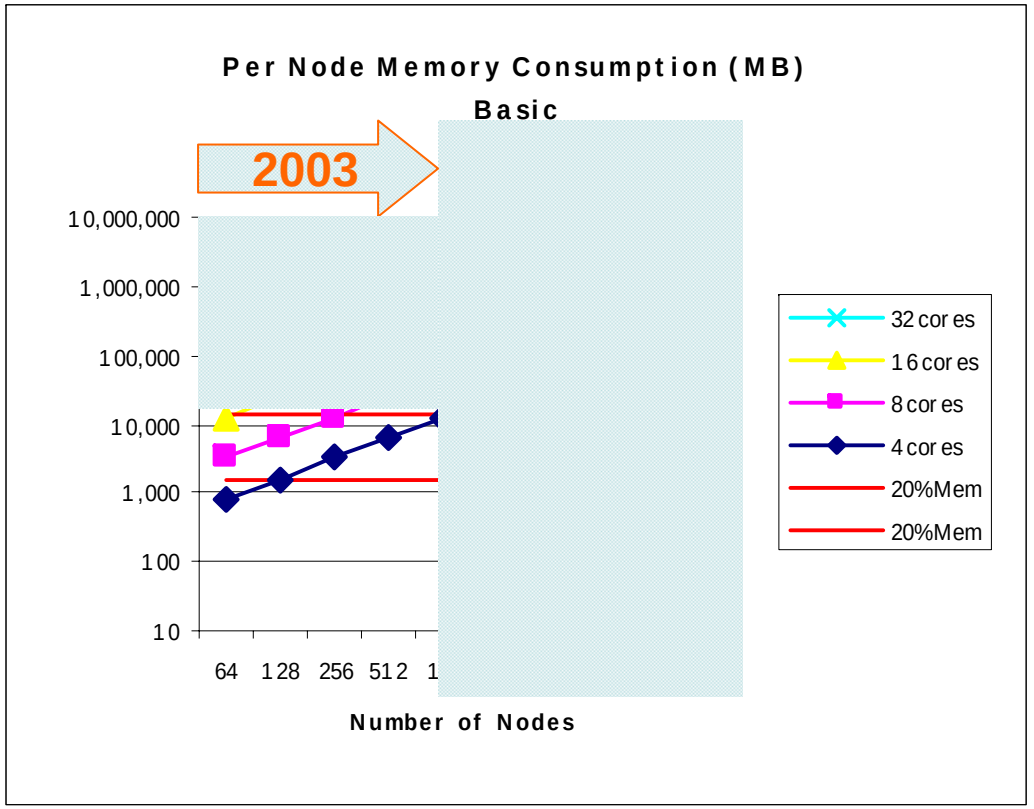
Agenda

- Scaling to Large Clusters
- Reliable Multicast
- Stateless TCP/IP Offloads

Scaling to Large Clusters

- Major trends
  - Cluster size 10,000s nodes
  - CPU cores 16 and up
  - Memory ~2GB/core
- QPs per node
  - QPs = Nn x Nc^2
- Memory per QP
  - WQ buffer ~32KB
  - RQ data buffers ~768KB

  * Ballpark numbers

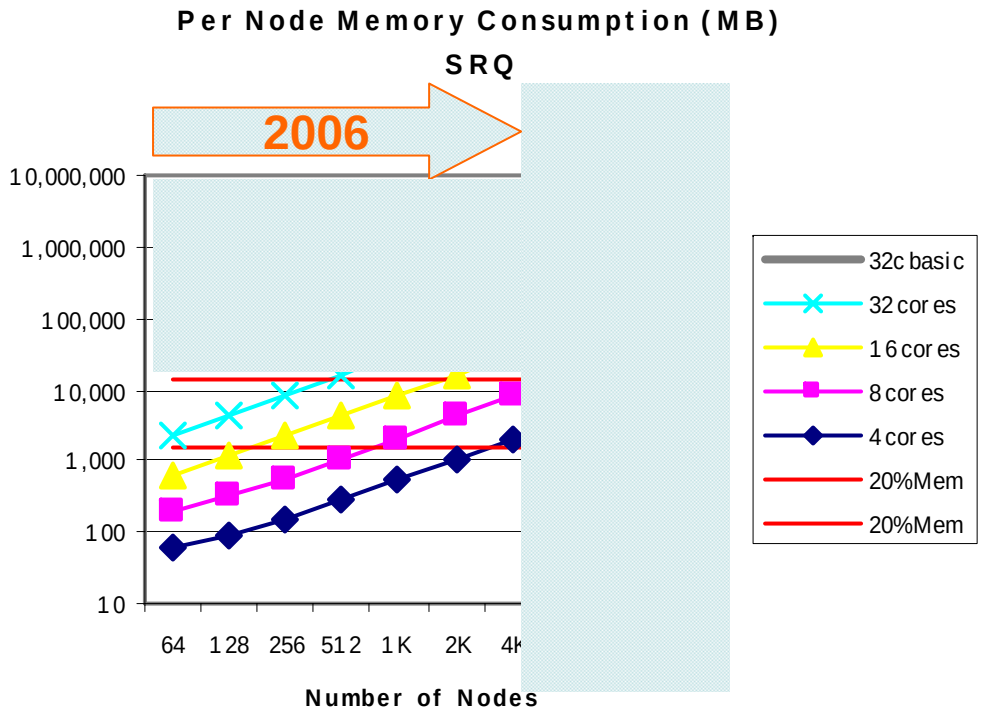


- To scale beyond ~1K nodes / 4 cores - Shared Receive Queue (SRQ) !

Nc = CPU Cores/Node
Nn = Nodes

Scaling with SRQ

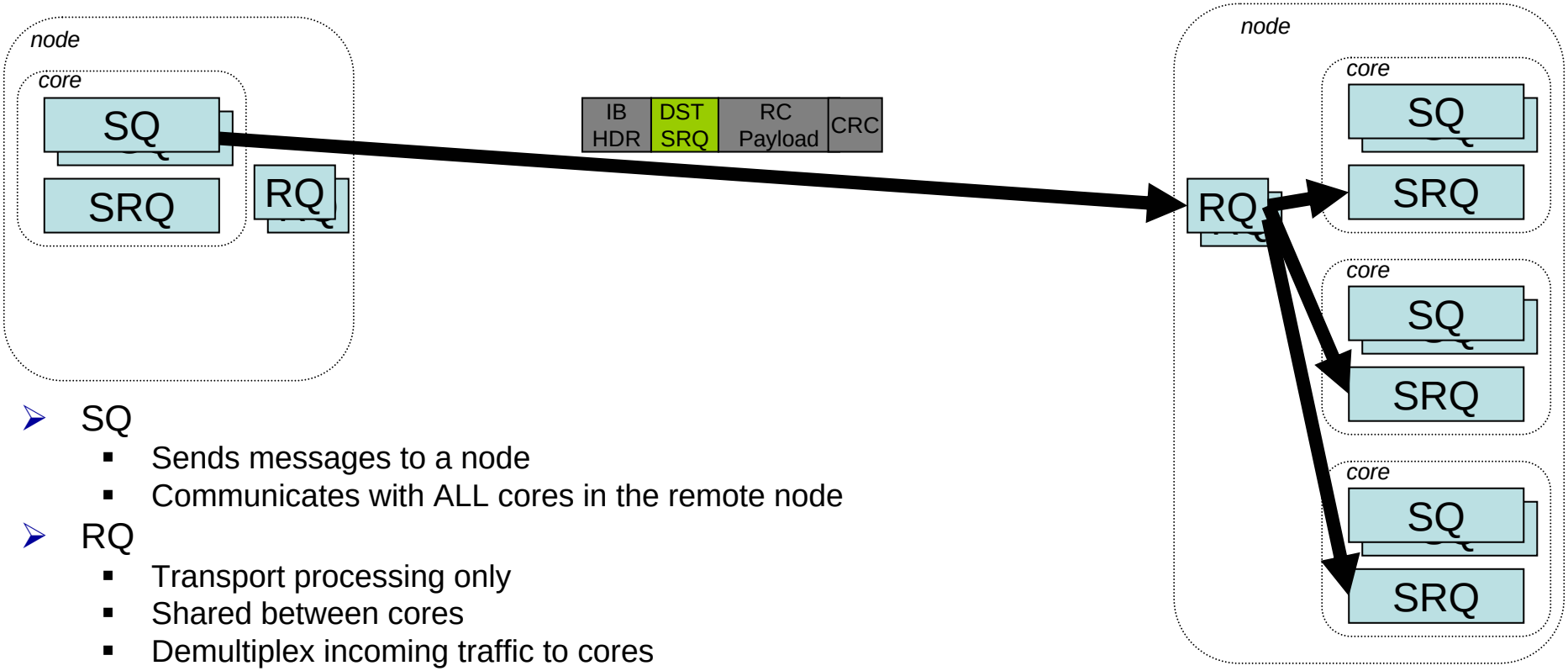
- QPs per node
  - QPs = Nn x Nc^2
- Memory per QP
  - SQ WQ buffer ~32KB
- Memory per core
  - SRQ data buffers ~7680KB



- To scale beyond ~4K nodes / 4 cores - Scalable Reliable Connected

Nc = CPU Cores/Node
Nn = Nodes

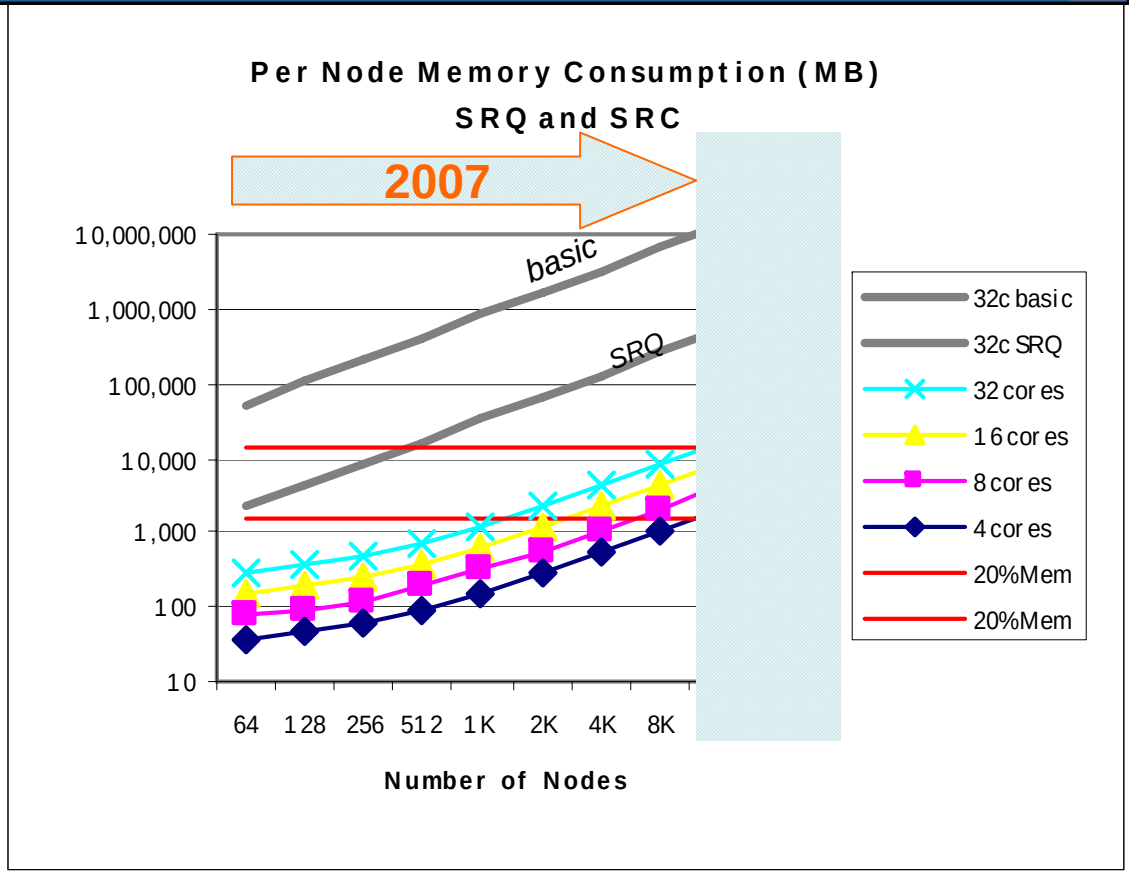
Scalable Reliable Connected



- SQ
  - Sends messages to a node
  - Communicates with ALL cores in the remote node
- RQ
  - Transport processing only
  - Shared between cores
  - Demultiplex incoming traffic to cores
    - Sends – take WQEs, CQ and PD from SRQ
    - RDMA Read/Write – take PD from SRQ
- Similar idea to IBTA RD

Scaling with SRQ and SRC

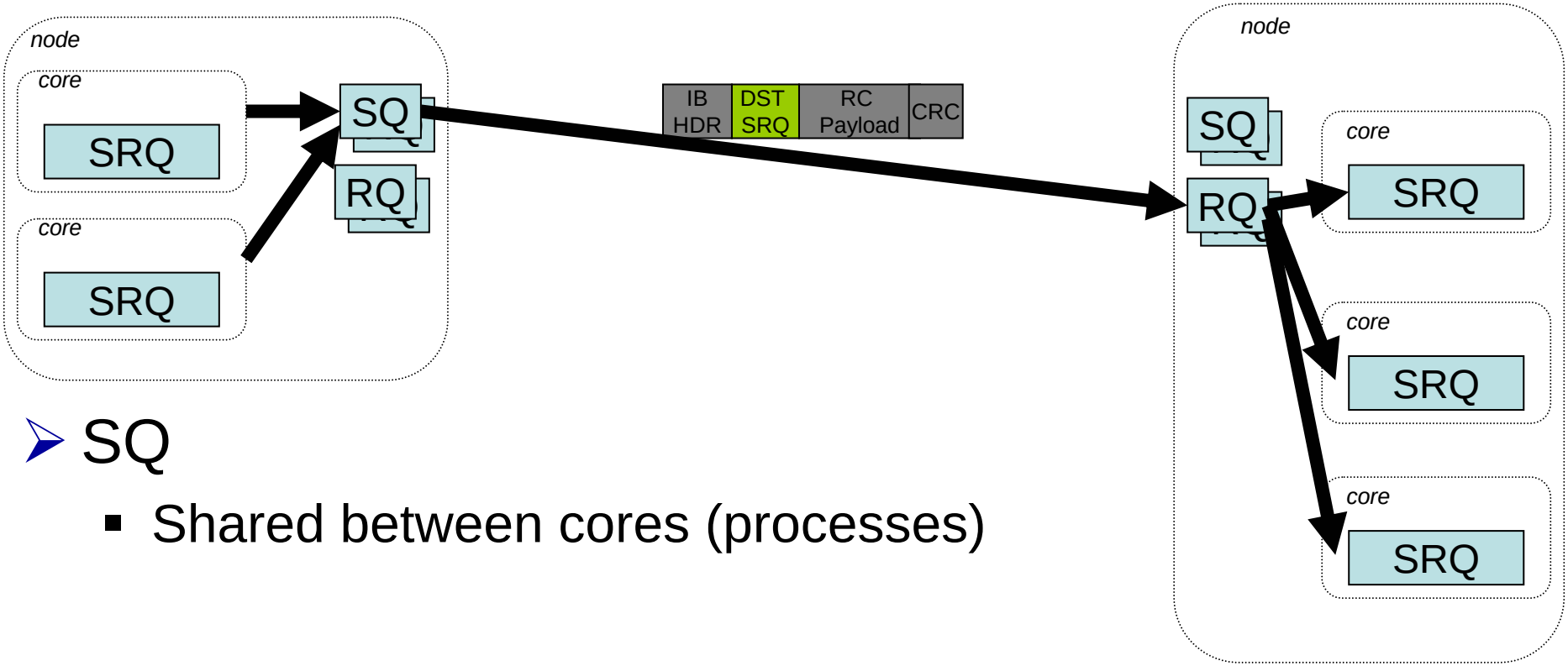
- QPs per node
  - QPs = Nn x Nc
- Memory per QP
  - WQ buffer ~32KB
- Memory per core
  - SRQ data buffers ~7680KB



- To scale beyond ~10K nodes / 16 cores
  – Shared Send Queue (SSQ)

Nc = CPU Cores/Node
Nn = Nodes

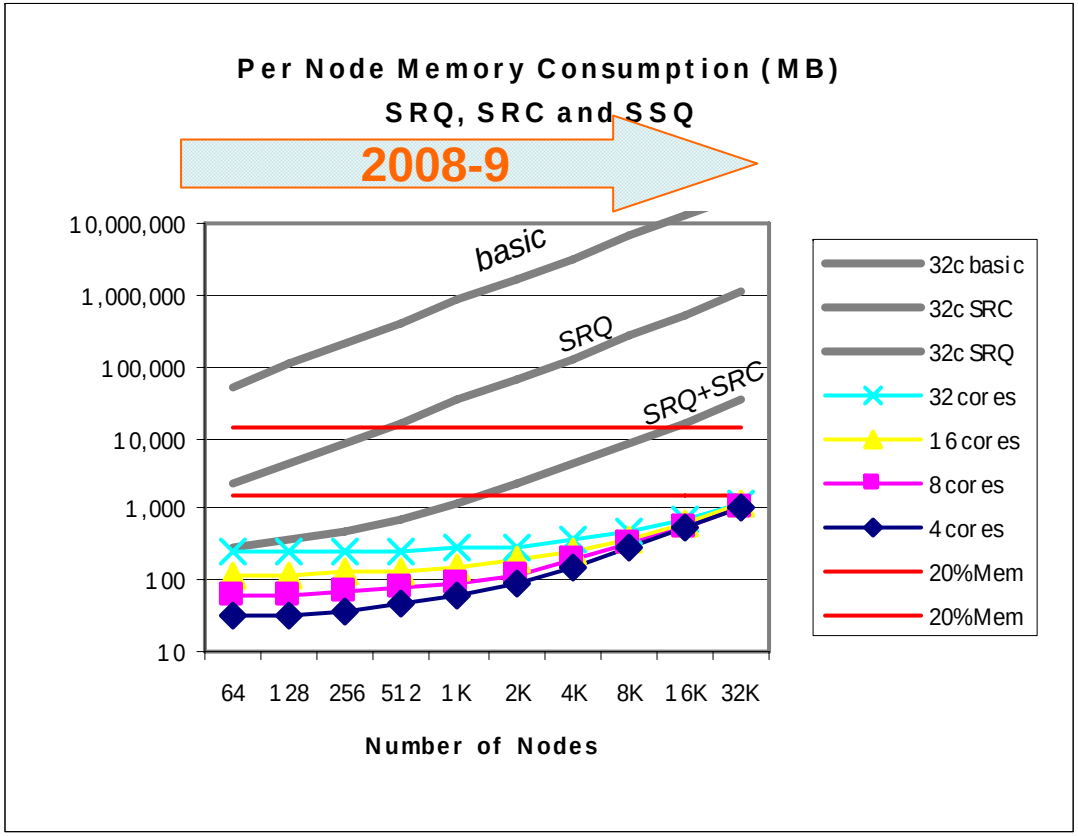
Shared Send Queue (SSQ)



- SQ
  - Shared between cores (processes)

Scaling with SRQ, SRC and SSQ

- QPs per node
  - QPs = Nn
- Memory per QP
  - WQ buffer ~32KB
- Memory per core
  - SRQ data buffers ~7680KB
- Can scale independently of the number of cores !



- Let's work out the details…

Nc = CPU Cores/Node
Nn = Nodes

SRC API Suggestion

- Query device for SRC support
- Attach or create an SRC domain
  - Can use shmget (2) as a reference
- Create an SRQ
  - SRQ is affiliated with the SRC domain
  - SRQ points to a CQ
- Core i in each node connects to core i in all remote nodes
  - QP type is SRC
  - Nn-1 QPs per core
  - QP is affiliated with the SRC domain
  - RQs become shared automatically (across all QPs with same SRCD)
- Send WQE
  - Adding destination SRQ
- SRC Domain Release
  - Like shmdt (2)

* Userland API only

Nc = CPU Cores/Node
Nn = Nodes

SSQ API Suggestion

- Query device for SRC **and SSQ** support
- Attach or create an SRC Domain
  - Can use shmget (2) as a reference
- **Attach or create a QP Domain**
  - **Enables sharing a QP**
- Create an SRQ
  - SRQ is affiliated with the SRC domain
  - SRQ points to a CQ
- Core **0** in each node connects to core **0** in all remote nodes
  - QP type is SRC
  - Nn-1 QPs per core **0**
  - QP is affiliated with the SRC domain
  - RQs become shared automatically (across all QPs with same SRCD)
  - **QP is associated with QP Domain (to be shared)**
  - **QP objects to be stored on a shared memory chunk**
- **Core i (other than 0) create a stub QP**
  - **Use QP Domain – indicates QP sharing**
- Send WQE
  - Adding destination SRQ
- SRC Domain **and QP Domain** Release
  - Like shmdt (2)

* Userland API only

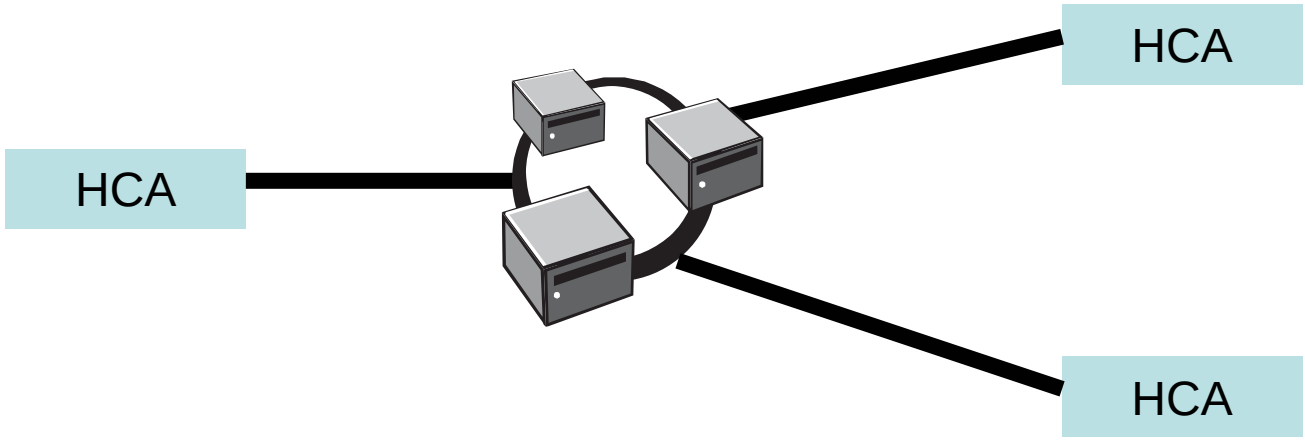
Nc = CPU Cores/Node
Nn = Nodes

Agenda

- Scaling to Large Clusters
- Reliable Multicast
- Stateless TCP/IP Offloads

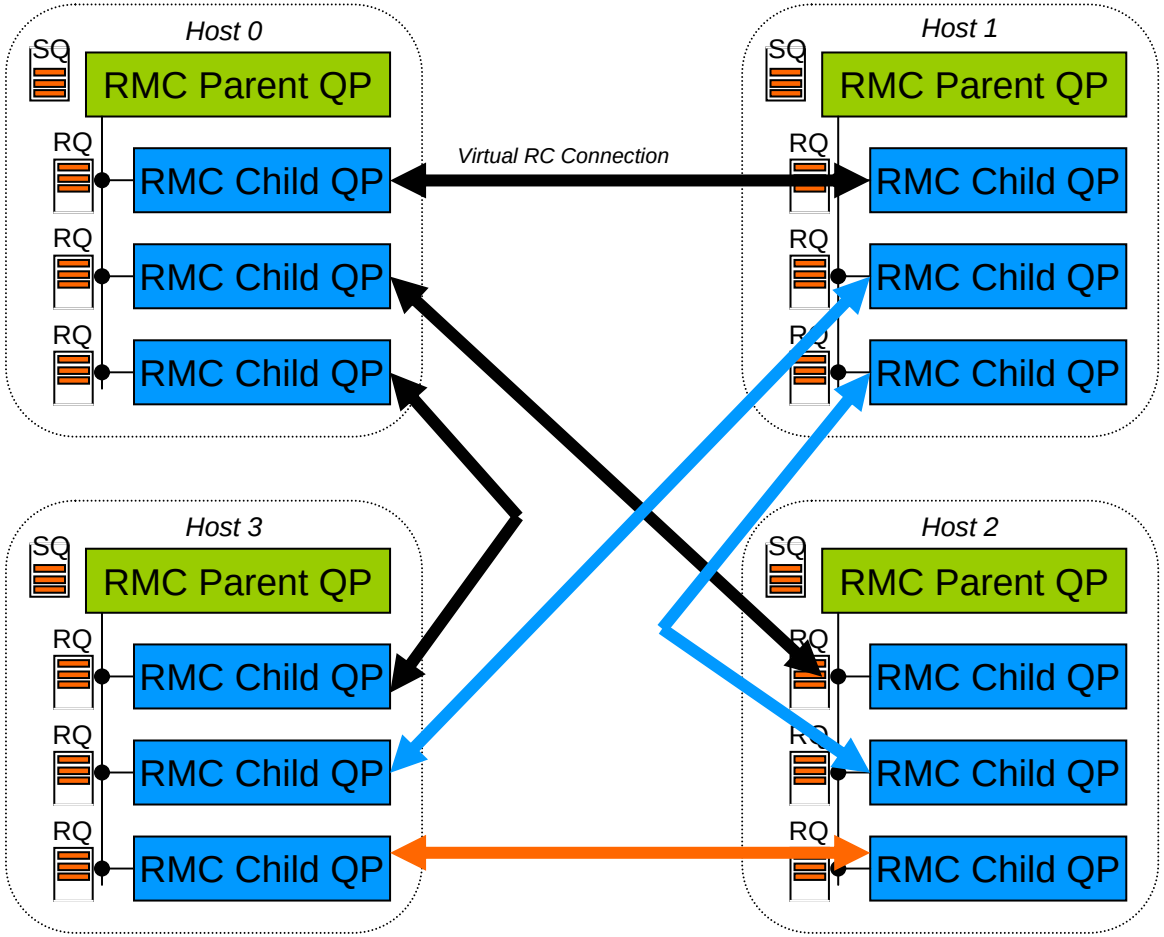
Reliable Multicast (RMC)

- RMC
  - HW Mechanism
  - Reliability guarantee
  - Beyond single MTU message multicast offload
- Applications
  - Distributed analysis of massive amounts of data
  - Scaling online trading, live news and video distribution
  - Speeding up of high performance MPI collective operations



RMC Control Objects

- RMC Parent QP
  - Owns the real SQ
  - Owns the MCG
    - MGID for each tree
  - Reports SEND completion
  - Governs the children
- Child QP
  - Virtually connected
  - Manages ACK transport
  - Reports RECV completions



RMC API Suggestion

- Query device for RMC support
- Create RMC Parent QP
  - QP owns the SQ (no RQ)
  - Attach to MCG (send only)
- Create RMC Child QPs
  - QP is associated with the parent
  - QP owns the RQ or SRQ (no SQ)
  - One QP per remote peer (say peer i)
  - Virtually connect QP to peer i
    - To allow sending back acknowledges
    - PSNs should be adopted from parent
  - Attach each QP to peer i MCG
- Send
  - Through ibv_post_send() on parent QP (no change in API)
- Receive
  - Sends are scattered on Child QP receive queue (no change in API)

* Userland API only

Agenda

- Scaling to Large Clusters
- Reliable Multicast
- Stateless TCP/IP Offloads

IPoIB Stateless TCP/IP Offloads

- Checksum
- TSO
- Receive Core Affinity
- Interrupt Moderation

* Kernel Only

Checksum Offload

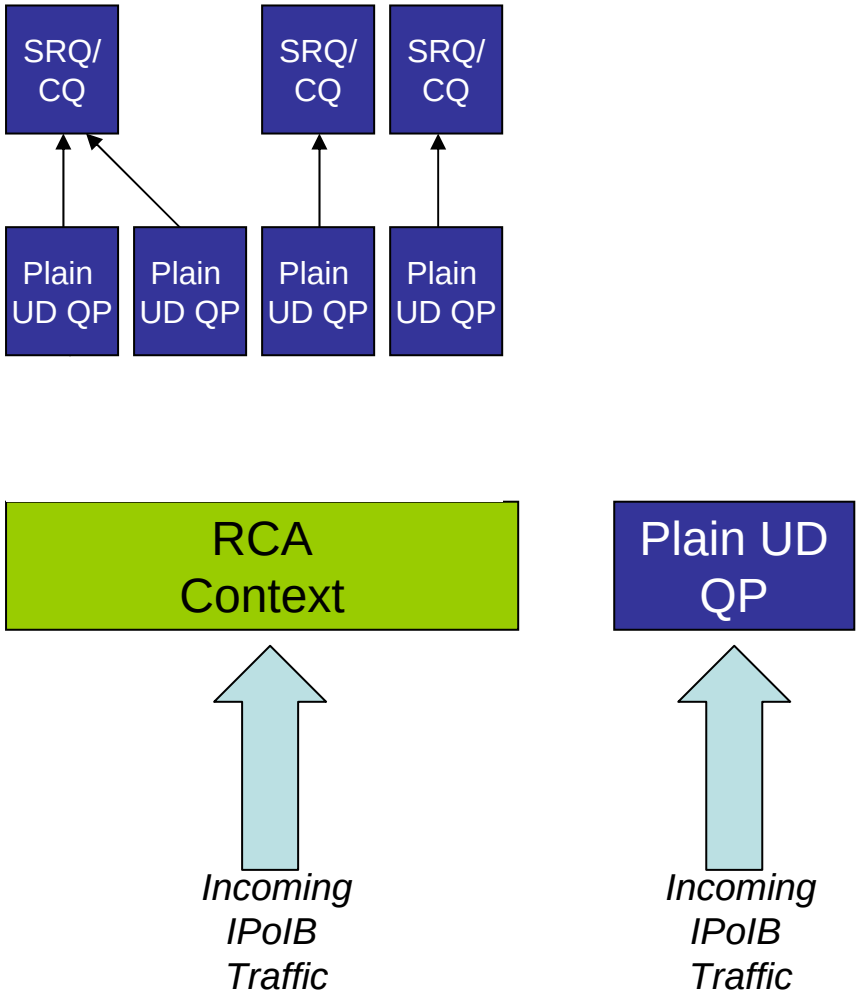
- TCP/UDP/IP Checksum Offloading
  - Query device for checksum offload support
- QP Creation
  - Mark QP for IPoIB checksum support
- TX
  - ibv_send_flags indicate checksum offload request
- RX
  - ibv_wc_flags indicate checksum status (good, bad, unverified)

TSO

- TCP/IP Segmentation Offload
  - Query device on TSO support
- QP Creation
  - Mark QP for IPoIB TSO support
- TX
  - ibv_send_flags indicate TSO request
  - MSS specified in new field
    - Can overload imm
  - 1st s/g element specifies the TCP/IP header

Receive Core Affinity (RCA)

- Query device for RCA support
- RCA Context QP Creation
  - Mark QP for RCA support
  - QP points to a chunk of QPs to distribute incoming traffic (should be a contiguous chunk of QPs)
- Modify QP
  - Enables changing SRQ/CQ affiliation for each QP (rebalancing)



Interrupt Moderation

- Per CQ/EQ/Interrupt/MSI settable moderation thresholds

Thank You !