Dynamically Connected Transport

Richard Graham
New Transport

- **Challenges being addressed:**
  - Scalable communication protocol
  - High-performance communication
  - Asynchronous communication
  - Reliable Transport

- **Current status: Transports in widest use**
  - RC
    - High Performance: Supports RDMA and Atomic Operations
    - **Scalability limitations:** One connection per destination
  - UD
    - Scalable: One QP services multiple destinations
    - **Limited communication support:** No support for RDMA and Atomic Operations, unreliable

- **Need scalable transport that also supports RDMA and Atomic operations ➔ DC** – The best of both worlds
  - High Performance: Supports RDMA and Atomic Operations, Reliable
  - Scalable: One QP services multiple destinations
IB Reliable Transports Model

- QoS/Multipathing: 2 to 8 times the above
- Resource sharing (XRC/RD) causes processes to impact each other
The DC Model

• Dynamic Connectivity

• Each DC Initiator can be used to reach any remote DC Target

• No resources’ sharing between processes
  – process controls how many (and can adapt to load)
  – process controls usage model (e.g. SQ allocation policy)
  – no inter-process dependencies

• Resource footprint
  – Function of HCA capability
  – Independent of system size

• Fast Communication Setup Time
Connect-IB – Exascale Scalability

Host Memory Consumption (MB)

- InfiniHost, RC 2002
- InfiniHost-III, SRQ 2005
- ConnectX, XRC 2008
- Connect-IB, DCT 2012

- 8 nodes
- 2K nodes
- 10K nodes
- 100K nodes
Dynamically Connected Transport

- Key objects
  - DC Initiator: Initiates data transfer
  - DC Target: Handles incoming data
Reliable Connection Transport Mode
Dynamically Connected Transport Mode
DC Verbs

• New objects
  – DC Initiator (new QP type)
  – DC Target (ibv_dct)
    SRQ, CQ, and DC Access Key associated with target

• Query Device used to check for support
DC Initiator Verbs

• DC Initiator Creation
  – Use `ibv_create_qp_ex()` extended verb
  – Add new QP type: `IBV_QPT_DC_INI`
DC Target Verbs

• Create DC Target verb
  ```c
  struct ibv_dct *ibv_exp_create_dct(struct ibv_context *context, struct ibv_exp_dct_init_attr *attr)
  ```

• Destroy DC Target verb
  ```c
  int ibv_exp_destroy_dct(struct ibv_dct *dct)
  ```

• Query DC Target verb
  ```c
  int ibv_exp_query_dct(struct ibv_dct *dct, struct ibv_exp_dct_attr *attr)
  ```
DC Send Verb

• Modify the extended send verb
  – Add to work request description:
    
    ```c
    struct {
      struct ibv_ah *ah;
      uint64_t   remote dct_access_key;
      uint32_t   dct_number;
    }dc;
    ```
Receive verbs

• No API changes
• Completion notification to CQ associated with the DC Target
Thank You