IB ACM

InfiniBand Communication Management Assistant (for Scaling)

Sean Hefty

Problem

MPI job startup places a huge burden on the fabric

Need to resolve addresses

Need to resolve paths

Problem
Address Resolution

- ARP for address resolution
  - ARP storm as all nodes try to discover other nodes
    - Number of outstanding ARP requests = $O(n^2)$
    - May require pre-loading ARP cache before starting MPI job
  - ARP replies require path record queries!
  - ARP entries timeout across the subnet
    - 1000 nodes → 1 million ARP entries subnet wide
    - 15 minute timeout → 1000 entries timeout *per second*
    - 24 hour timeout → 12 entries timeout *per second*
    - Even with a 24 hour ARP entry timeout, 7200 entries will timeout during a 10 minute MPI run!

No easy solution, but workable in practice

Problem
Path Resolution

- Centralized SA hinders scalability
- Number of outstanding queries = $O(n^2)$
- Queries take *minutes* to complete
  - If apps actually wait that long
  - (Think of how many ARP entries just timed out!)
  - Responses are not shared

Standard path record query mechanisms simply do not work at scale

Solution
IB ACM

- Addresses connection setup scalability issues
- Service / daemon providing:
- Address resolution
  - ARP like protocol over IB multicast
  - Names may be host names, IP addresses
- Route resolution
  - Construct or query for path records
  - Path record caching

Solution
ACM for Address Resolution

- Wait, why duplicate ARP mechanisms?
- IPoIB issues path record queries as part of its ARP implementation
- Store address and route data together
  - Timeout data together
  - Under certain conditions, we can resolve address and route data without using SA queries
- We can obtain additional information about the remote device
  - E.g. maximum RDMA capabilities
- **Restrict the number of outstanding address resolution requests on the subnet to O(n)**

Solution
ACM for Route Resolution

- Is ACM a path record cache?
- Yes, but also an open source framework for addressing scalability issues
  - May be able to construct path records without querying the SA
- Integrates with the librdmacm to make its use transparent to the user
- **Restricts the number of outstanding SA queries to the subnet to O(n)**

Usage Model

- Recommended use is via librdmacm
  - Requires '--with-ib_acm' compile option
    - Temporary requirement, will be removed once ACM is more proven
  - Falls back to normal resolution on failures
    - Unable to contact local or remote ib_acm service
    - Failure to resolve address or path information
    - Small latency hit if librdmacm compiled with ACM support, but service is not running

Usage Model

- rdma_resolve_route
  - All existing librdmacm apps can take advantage of path record caching
  - Path data is obtained from ib_acm service and kernel is configured to use it
    - ib_acm lookup is synchronous
  - Requires kernel 2.6.33 (OFED 1.5.2) or newer

rdma_resolve_addr still ends up going through ARP :P

Usage Model

- rdma_getaddrinfo
  - Supports address and route resolution
    - Kernel support to make use of address resolution is pending (AF_IB patch set)
  - Can act as a simple (i.e. dumb) path record query interface
    - For apps that manually configure their QPs, but require path records (specifically SL data) from the SA to avoid subnet deadlock
  - NODELAY flag
    - Quick check against cached data
    - Kicks off ACM resolution protocols in background
    - Future lookups can find cached data

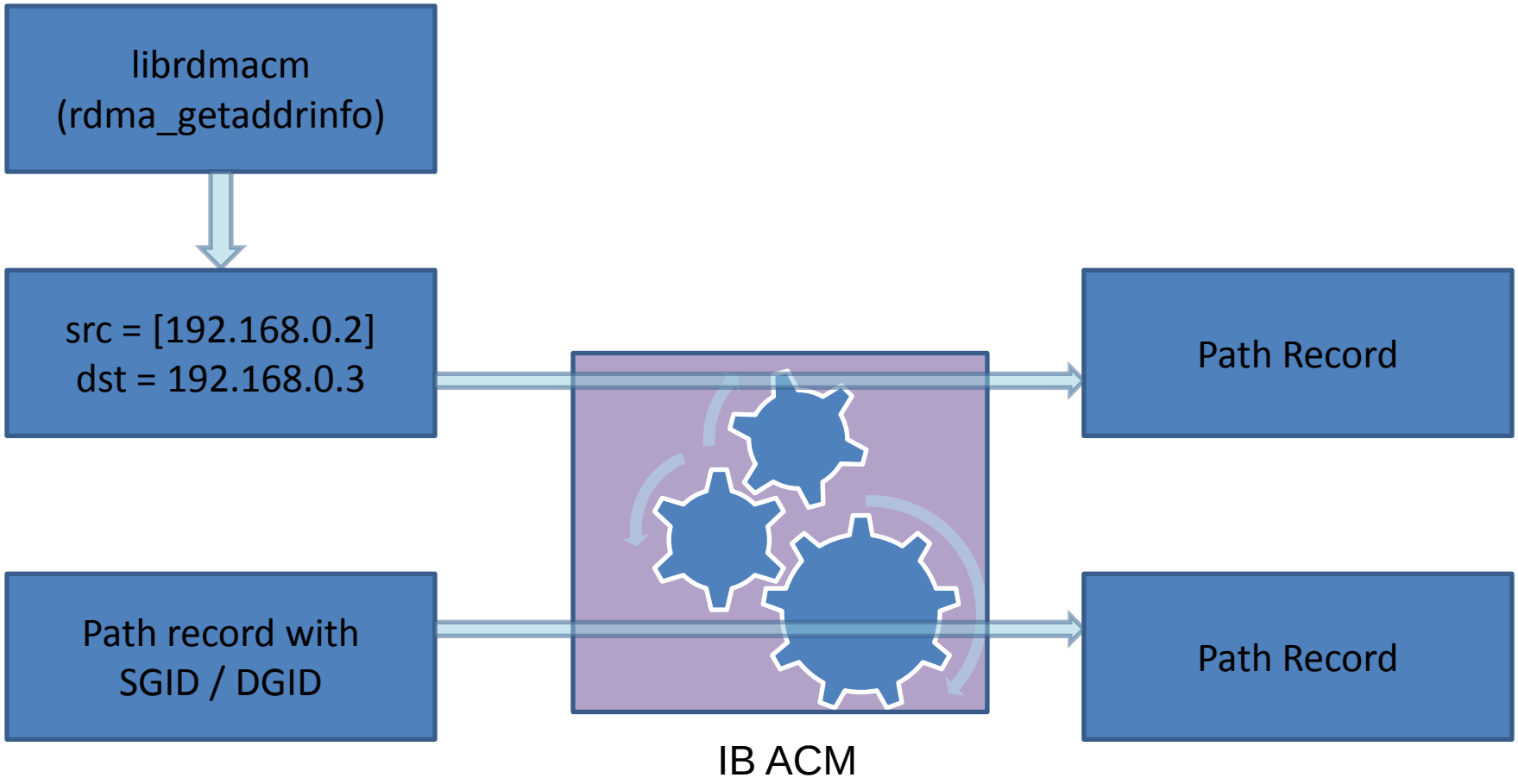
Usage Model

- ib_acm may be accessed directly via a socket interface
  - **Not recommended**
  - Protocol is defined by acm.h header file, but is not documented
  - librdmacm and ib_acme may be used as guides

ACM Service

- Listens on TCP socket
- Simple request / reply protocol
- Input:
  - Name, IP address, partial path record
  - Destination and optional source
- Output:
  - Set of path records
    - Forward, reverse, alternates, CM paths
  - Current implementation is 1
    - Path must be fully reversible

ACM Service Examples



ACM Operation

1. All ACM services join multicast groups
   - Groups differ by pkey, rate, MTU, type of service
   - All services must join 1 common group
   - All traffic occurs on common group
2. ACM receives client request
3. Initiator broadcasts destination address on common multicast group
4. All peer ACM services cache source data

ACM Operation

5. Target service queries SA or constructs path record to initiator
   - See next slide
6. Target service responds with IB address
7. Initiator caches response address
8. Initiator queries SA or constructs path record to target
9. ACM responds to client request

ACM Operation

- Query SA for path data
  - Required for certain fabric topologies to avoid deadlock
- Construct path record based on common multicast group supported by source and destination
  - Can be more efficient
  - Once multicast groups are configured, avoids querying the SA

ACM Communication

- Defines application specific MADs
  - Uses management class 44
- Allocates a UD QP
- Protocol is defined internal to ACM service
  - Defined in internal header files
  - Undocumented

ACME

- Multi-purpose ACM test and configuration application
- Verifies path data from ACM against SA path records
  - Can be used to validate if multicast resolution protocol is usable
- May be used to pre-load cache
- Generate configuration files

ACM Configuration

- Users may override default ACM configuration options through the use of configuration files
- Default files can be generated using the ib_acme utility
  - acm_addr.cfg
    - Contains local address data
    - File will be unique per node
  - acm_opts.cfg
    - Contains ACM service options
    - Likely the same across an entire cluster

ACM Configuration

- Address assignments
  - <device, port, pkey, name>
  - Name is usually host name or IP address
  - **Pkey can be use for QoS**
- Logging location and details
- Timeout / retry settings
- Outstanding requests limits
  - To SA or peer ACM services
  - Restricts the number of outstanding requests on the subnet

Work In Progress

- Validate concepts, performance, and scalability
  - Initial testing is positive
- Support for dynamic changes is limited
  - Handles port up/down events
  - Does not respond to local IP address or hostname changes
- Want to obtain path data progressively
  - Pre-load cache using ib_acme
  - Save / restore data between reboots

Work In Progress

- Cached data not invalidated
  - Must stop / restart service
  - Proposal is to invalidate data based on CM timeouts
    - Dependent on kernel changes
- QoS support controlled by SA
  - Addresses / names map to specific pkey values
  - A QoS scheme based on pkeys would work best

Pending Work
AF_IB Support

- Allows direct IB addressing with transport independent rdma_getaddrinfo interface
- Desired for librdmacm to make use of ACM address resolution
- Removes librdmacm requirement for ipoib
- Provides greater flexibility moving forward to support alternate paths

Pending Work
MAD Snooping

- Invalidate cache based on local events
  - CM timeouts, rejects
- Avoid registering for SA events
  - Data is invalidated based on local events only
- Capture path data from other applications
  - ipoib, srp, iser, rds, etc.