Scalable name and address resolution infrastructure

-- Ira Weiny/John Fleck

#OFADevWorkshop
SA interaction difficulties

- SA MAD formats, RMPP, libibumad “quirkiness”
- Application shortcuts
  - Hard coded PR data
  - Ignoring parts of queried PR data
  - Only work on a limited set of clusters or cluster types
- Direct access to libibumad and the SA are vectors for security breaches
Current stack is not scalable

- Nodes access the same SA services multiple times from ibacm, kernel, libibumad…
  - PR queries
  - Notice/multicast registrations
- Name resolution through standard DNS requires an ARP from IP to GID
ibacm name resolution

• Relies on IPoIB (DNS, ARP, etc)
• Names map to <GID, Pkey> “end point”
  – User’s often don’t care about the partition they are running on.
  – “cross” partition names can’t be resolved
    • Local apps need knowledge of a common partition prior to name resolution.
  – Some work done in this area via ibacm_hosts.data
• Current name resolution requires source “end point” to be specified
ibacm as a SA proxy

• ibacm provides a good starting point for addressing some of these concerns…
Goals

• Provide controlled and consistent access to user space name and PR resolution services (AKA SA access)
  – SA access control
  – Accuracy
  – Ease of use
  – Portability
  – Enable all consumers to access ibssa

• Provide caching and other ibacm services to kernel users
ibacm enhancements

• Applications query ibacm as a local “SA proxy”
  – All SA interactions done through ibacm
  – Additional name services provided
  – ibacm can control access to SA and libibumad

• ibacm is backed by “providers”
  – ibssa
  – Current features as default
  – Enhancements for name services are planned
ibacm enhancements

• Name resolution services
  – “DNS” for direct name resolution
    • Name to PR (or GID, <GID, PR>, IP, <IP, PR>)
• ibacm provides service to the kernel
  – Uses netlink
  – Leverages the same infrastructure for all users
Architecture (non-eye chart)
Implementation plans

• Separate out ibacm into “core” and default provider
• Core handles
  – Provider loading and assignment to ports/End points
  – Steering client requests to correct provider
  – Port/device Events
  – Netlink requests and events
  – Administration like config file parsing, log file, etc
• Default provider handles
  – Same functionality as current resolve functions
Initial data flow details

1. Request('resolve', src, dest)
2. ibacm core
   select provider
3. Resp(PathRecord)
4. Resp(PathRecord)

Ibacmp (Default provider) -> provider 1
provider 2
Provider API’s

• Prototype code being worked
  – Collaborating with OFI WG and rdmacm
  – submission to the list imminent
  – “prov” branch in ibacm’s git tree

• The API will evolve, collaborating with ibssa

• Main API calls will include
  – Path Record resolution
  – Name to GID mapping helper
Expand *__getaddrinfo

- Use ibacm first to resolve a Name prior to calling getaddrinfo (DNS)
  - Call can provide Path Record hints through the normal “hints” parameter
    - For example Service ID, Pkey etc.
- Need both librdmacm and ibacm changes
- Only single local end point can be supported now
  - Future local end point resolution can be determined by GID returned from provider name -> GID map
Kernel ibacm access

• SRP, IPoIB, and rdma_cm kernel modules use ib_sa to query for Path Records
• Extend ibacm PR resolution/caching to kernel modules
• Use netlink messages to communicate with ibacm
• Expand existing RDMA netlink interface
• Currently connecting with ib_sa using ibacm messages
  – Exploring the use of ib_mad and using MAD formatted messages
A little more detail…
# Current Netlink / ibacm Message Format

<table>
<thead>
<tr>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Length</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flags</td>
<td>Type</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sequence</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pid</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data[0]</td>
<td>Status</td>
<td>Opcode</td>
<td>Version</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tid</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flags</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td>Type</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Path Record (64 Bytes)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Byte Order:
- Netlink Header (16 bytes)
- ACM Header (16 bytes)
- Path Data (72 bytes)
Kernel still uses SA when ibacm not available
Future work

• SA event registration and reporting
  – Notice

• Multicast

• IP to GID mapping
  – IPoIB netlink to ibacm?
  – arpd extentions?
Thank you

- Hal Rosenstock
- Kaike Wan
- Sean Hefty
Thank You
Current SA interactions

- **Applications**
  - Direct SA
    - Libibumad
    - UD QP
  - Librdmacm
    - Ibacm
    - Dns/arp

- **Kernel**
  - Direct SA access only
Name service requirements

• Generic interface to request remote node by name through “DNS like” resolution
  – Mapping provided by providers based on cluster configuration, node configuration, and/or provider/SA communication.
librdmacm example

- New librdmacm example app

```bash
$ resolve_name -h
```

Usage: `resolve_name <name>`

- [-h]
- [-s <service id>] Specify a service ID in PR 'hints'
librdmacm example

$ resolve_name priv03
ai_family 0
ai_route : 0x1ff15a0
Path information
  service_id: 0x0
dgid: fe80::11:7500:79:1815
sgid: fe80::11:7500:79:1763
dlid: 2
  slid: 1
...

$ resolve_name google.com
ai_family 2
dest (null) 173.194.33.133