**Overview**

- What is CCI?
  - A generic, communication abstraction layer
  - Platform for experimentation and production
  - Supports a number of networks (SAN, LAN, WAN)
    - InfiniBand, Gemini, Ethernet, etc.
  - Modern semantics provide fine grained control over individual flows
  - Semantics enable high-performance in the face of out-of-order delivery
    - No need to guarantee order within a single RMA for bulk data movement
  - New features
    - Completed initial WAN optimized CCI transport
    - Adding support for “virtual fabric support” (CCI Routed)
    - Provides end-to-end virtual fabric over heterogeneous networks

---

**Key Characteristics**

- Connection
  - Provides a channel to communicate with a peer
  - Allows the application to select the required level of QoS in terms of reliability and ordering (i.e., unreliable-unordered, reliable-ordered, reliable-unordered)
- Two types of communication paradigms
  - Two-sided: send/recv
  - One-sided: Remote Memory Access (RMA) read/write
- Endpoints
  - Virtual instances of a device,
  - Enables better scalability by limiting the amount of resources allocated to communications with a given peer
  - A single endpoint can handle many connections
- Events
  - All connection and communication operations return immediately
  - Application can poll the endpoint for completion events

---

**CCI for HPC**

- CCI-aware applications can implicitly use high-performance interconnects
  - Implicitly improve latency and throughput
- Improve application portability
  - CCI abstracts the underlying network
  - CCI support all major HPC interconnects
- Example: CCI over Cray Gemini
  - CCI latency higher because of copy-out on the receiver, locking for multi-threaded applications and progressing the connection socket

**CCI for the WAN**

- CCI provides tunable transports for the WAN
  - TCP, UDP, Verbs (RoCE and iWarp)
- Support routing for end-to-end communication over heterogeneous networks (across both LAN and WAN)
- Example: CCI over 10Gb/s Ethernet between ANL and NERSC (DOE ESnet ANI)

---

**Persistent Services**

- HPC goes beyond MPI, some services exist outside of scheduler jobs or span multiple jobs (persistent services)
  - Distributed file systems
  - Code coupling
  - Health monitoring
  - Debugging
  - Performance monitoring
  - Master/slave communication model
- Current solutions
  - Use Sockets: highly portable but limited performance
  - Implement their own network abstraction layer: efficient but expensive in terms of development
  - CCI can simplify support persistent services
  - CCI provides a common NAL
  - CCI provides improved performance compared to Sockets

**Bulk-synchronous Communication**

- MPI over CCI
- Native CCI transport in Open MPI available
  - Developed by UTK
  - Allows quick prototyping and testing of complex use-cases
- Example
  - Two MPI applications doing connect/accept
  - 10Gb/s link between ANL and NERSC (DOE ESnet ANI)
  - Split a big file into blocks among ranks
  - Parallel file access using POSIX API
  - Non-blocking parallel data transfer using MPI_Isend() and MPI_Irecv()

<table>
<thead>
<tr>
<th>Block Size</th>
<th>1 GB File (throughput in MB/s)</th>
<th>10 GB File (throughput in MB/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 MB</td>
<td>488.17</td>
<td>1012.42</td>
</tr>
<tr>
<td>10 MB</td>
<td>481.71</td>
<td>963.52</td>
</tr>
</tbody>
</table>

Contact: Geoffroy R. Vallée (valleege@ornl.gov)