USING OPEN FABRIC INTERFACE IN INTEL® MPI LIBRARY

Michael Chuvelev, Software Architect

Intel

April 11, 2018
INTEL MPI LIBRARY

- **Optimized MPI application performance**
  - Application-specific tuning
  - Automatic tuning
  - Support for latest Intel® Xeon® Processor (codenamed Skylake) & Intel® Xeon Phi™ Processor (codenamed Knights Landing)
  - Support for Intel® Omni-Path Architecture Fabric, InfiniBand*, and iWarp and RoCe Ethernet NICs, and other Networks

- **Lower latency and multi-vendor interoperability**
  - Industry leading latency
  - Performance optimized support for the fabric capabilities through OpenFabrics*(OFI)

- **Faster MPI communication**
  - Optimized collectives

- **Sustainable scalability**
  - Native InfiniBand* interface support allows for lower latencies, higher bandwidth, and reduced memory requirements

- **More robust MPI applications**
  - Seamless interoperability with Intel® Trace Analyzer & Collector

Learn More: software.intel.com/intel-mpi-library
Libfabric Enabled Applications

Libfabric
- Control
  - Discovery
- Communication
  - Connection Mgmt
  - Address Vectors
- Completion
  - Event Queues
  - Counters
- Data Transfer
  - Message Queues
  - RMA
  - Tag Matching
  - Atomics

OFI Provider
- Discovery
- Connection Mgmt
  - Address Vectors
- Event Queues
  - Counters
- Message Queues
  - RMA
  - Tag Matching
  - Atomics

NIC
- TX Command Queues
- RX Command Queues

Learn More: https://ofiwg.github.io/libfabric/
OFI ADOPTION BY INTEL MPI

- Year 2015: Early OFI netmod adoption in Intel MPI Library 5.1 (based on MPICH CH3)
- Year 2016: OFI is primary interface on Intel® Omni-Path Architecture in Intel MPI Library 2017
- Year 2017: OFI is the only interface in Intel MPI Library 2019 Technical Preview (based on MPICH CH4)
- Year 2018: Intel MPI Library 2019 Beta over OFI
  - Intel OPA, Ethernet*, Mellanox*; Linux*/Windows*
  - vehicle for supporting other interconnects
OFI FEATURES USED BY INTEL MPI

• Connectionless endpoints FI_EP_RDM
  • NEW: Scalable endpoints fi_scalable_ep() for efficient MPI-MT implementation
• Tagged data transfer FI_TAGGED for pt2pt, collectives
  • msg data transfer FI_MSG is a workaround if FI_TAGGED not supported
• RMA data transfer FI_RMA for one-sided; FI_ATOMICS
• Threading level FI_THREAD_DOMAIN
• FI_MR_BASIC/FI_MR_SCALABLE
• FI_AV_TABLE/FI_AV_MAP
• RxM/RxD wrappers over FI_EP_MSG/FI_EP_DGRAM endpoints and FI_MSG transport
  • for verbs, NetworkDirect support
Scalable Endpoints allow almost the same level of parallelism on multi-threads vs. multi-processes with much less AV space.

Intel MPI leverages per-thread (thread-split) communicators tied to distinct TX/RX contexts of Scalable Endpoints and distinct CQs.

Intel MPI expects a hint from user that application satisfies thread-split programming model.

With the hint, it safely avoids thread locking while the provider may use FI_THREAD_ENDPOINT/FI_THREAD_COMPLETION level wrt MT optimization.
CommA, CommB are associated with different RX/TX contexts, different CQs

As long as different threads don't access the same Scalable Endpoint context, they can be accessed in a lockless way with a provider supporting just FI_THREAD_ENDPOINT, or FI_THREAD_COMPLETION level.
POTENTIALLY USEFUL OFI EXTENSIONS

- **Collectives API:**
  - Expose HW based collectives via OFI
  - Implement collectives on the lowest possible level for SW based collectives
  - Enrich multiple runtimes with easy collectives

- **Collectives (Tier 1)**
  - handle = `/Coll(ep, buf, len, group, ..., sched = NULL)`
  - Test(handle) / Wait(handle)
  - need 'group' concept introduction [and datatypes/ops for reductions]
  - might have optional 'schedule' argument describing the algorithm

- **Schedules (Tier 2)**
  - handle = `/Sched_op(ep, buf, len, group, ..., sched)`
  - Test(handle) / Wait(handle)
  - graph-based algorithm for a collective operation (or any chained operation)
THANK YOU
Michael Chuvelev, Software Architect
Intel