OPEN MPI ON FRONTIER WITH LIBFABRIC

Amir Shehata, Systems Engineer

Oak Ridge National Lab
Open MPI on Frontier with Libfabric

Amir Shehata
shehataa@ornl.gov
Agenda

• Problem Statement & Goals
• Technical Landscape
• Solution Overview
• LINKx Provider
  • Shared Queues
• SHM Work
• Preliminary Performance Data
• Status
• Future Work
Problem Statement & Goals

• Problems addressed:
  - Vendor only provides Cray MPI on Frontier
  - Users need more choices of MPI implementations
    - work around problems
    - try out new features

• Goals:
  - Provide an alternative MPI using Open MPI
  - Provide comparable performance to Cray MPI
Technical Landscape

• Cray supports Slingshot 11 via a new CXI libfabric provider

• Three potential Solutions to use the CXI provider
  
  • Open MPI MTL Path
    • Use libfabric tagged message interface
  
  • Open MPI BTL Path
    • Use MPI for tag matching and other higher level logic, and libfabric for byte transfer only.
  
  • Open MPI UCX Path
    • Use UCX and integrate libfabric under the UCX API

• Open MPI MTL option was selected
Solution Overview

- Application
- OMPI
- libfabric
- LINKx
- SHM endpoints
- CXI endpoints
- OTHER endpoints

Write/read Process Information

Network/Memory send/recv
LINKx Provider

- LINKx is a new provider designed to link multiple providers
- Allows Open MPI to use libfabric for both local and remote communication
- It will make a decision based on peer locality, which provider endpoint to use
- LINKx will share both its completion queues and receive queues to reduce communication and memory overhead
- It can potentially be expanded to handle Multi-Rail
SHM Memory Work

- To support all MPI use cases the following SHM features have been added
  - Full support for ROCM HSA APIs
  - Added Asynchronous ROCM IPC Support
  - Added IPC Caching mechanism
  - Added XPMEM Support
    - XPMEM allows mapping remote process memory space locally. This provides an efficient method of sharing memory
  - Support XPMEM export for specific memory regions instead of exporting entire address space.
  - Added ROCM HIP API support (intended as reference implementation)
  - Support H2D via XPMEM, since XPMEM maps remote process memory locally, it can then be copied directly into Device memory
Collective Improvement

- Some key performance updates helped bring OMPI performance closer to Cray MPI Performance
  - Selection of the optimal interface to bind to a process
  - SHM locking improvements
    - SHM provider locking was very coarse, causing serialization between processes
    - Moved to more lock free strategy to minimize serialization
One-Sided Support

- Support One-Sided through LINKx to have a uniform Open MPI configuration for all possible use cases.
- Support scalable endpoints in LINKx
- Support RMA APIs in LINKx
- Support Atomics APIs in LINKx
- More work still to needs to be done.
INTRA-NODE pt2pt Performance

D2D bidirectional Bandwidth

H2H bidirectional Bandwidth
INTER-NODE pt2pt Performance

D2D bidirectional Bandwidth

H2H bidirectional Bandwidth
INTER-NODE Collective Performance (512np + 8ppn + 64n)

H2H All to All latency
INTER-NODE Collective Performance (512np + 8ppn + 64n)

H2H broadcast latency
Status

- **Completed**
  - LINKx with tagged message support
  - Shared completion and receive queues
  - SHM features (ROCM support, Asynchronous IPC, etc)
  - SHM locking improvements
  - OMPI Process/NIC Binding
  - Upstreamed ROCM support patches and lock optimization patches

- **In Progress**
  - Full LINKx implementation, non-tagged, atomic, etc
  - Performance Improvement
Future Work

- Collective performance improvements
- Complete one-sided support
- Support Intel GPUs on Aurora
- Complete LINKx support for all libfabric APIs
- Productize LINKx and test linking multiple providers
- Multi-Rail support via LINKx
- Upstream changes
Questions?