STATUS OF OPENFABRICS INTERFACES (OFI) SUPPORT IN MPICH

Yanfei Guo, Assistant Computer Scientist

Argonne National Laboratory
OVERVIEW

- What is MPICH?
- Why OFI?
- Current Support
  - MPICH 3.4 series (CH4)
  - MPICH 4.0a1 series (CH4)
- Future Plan
WHAT IS MPICH?

- MPICH is a high-performance and widely portable open-source implementation of MPI
- It provides all features of MPI that have been defined so far (up to and include upcoming MPI-4.0)
- Active development lead by Argonne National Laboratory and University of Illinois at Urbana-Champaign
  - Several close collaborators who contribute features, bug fixes, testing for quality assurance, etc.
    - IBM, Microsoft, Cray, Intel, Ohio State University, Queen’s University, Mellanox, RIKEN AICS and others
- Current stable release is MPICH-3.4
- Latest release is MPICH-4.0a1
- www.mpich.org
MPICH: GOAL AND PHILOSOPHY

- MPICH aims to be the preferred MPI implementation on the top machines in the world
- Our philosophy is to create an “MPICH Ecosystem”
MOTIVATION

Why OFI/OFIWG?

- Support for diverse hardware through a common API
- Actively, openly developed
  - Bi-weekly calls
  - Hosted on Github
- Close abstraction for MPI
  - MPI community engaged from the start
- Fully functional sockets provider
  - Prototype code on a laptop
- Strong Vendor Support
**CH4 device being the default**

- Replacement for CH3 as default option, CH3 still maintained till all of our partners have moved to CH4
- Co-design effort
  - Weekly telecons with partners to discuss design and development issues
- Three primary objectives:
  - Low-instruction count communication
    - Ability to support high-level network APIs (OFI, UCX)
    - E.g., tag-matching in hardware, direct PUT/GET communication
  - Support for very high thread concurrency
    - Improvements to message rates in highly threaded environments (MPI_THREAD_MULTIPLE)
    - Support for multiple network endpoints (THREAD_MULTIPLE or not)
  - Support for GPU
MPICH WITH CH4 DEVICE OVERVIEW

Application
MPI Interface
MPI Layer

Machine-independent Collectives
Derived Datatype Management (Yaksa)
Group Management

Abstract Device Interface (ADI)

CH4
CH4 Core
Architecture-specific Collectives
Active Message Fallback
GPU Support Fallback

Netmods
OFI  UCX

Shmmods
POSIX  XPMEM  GPU IPC

Legacy
CH3

© OpenFabrics Alliance
**Required Features for GPU Support**

- Using GPU memory in intra-/inter-node communication
- Supporting MPI features such as datatype, collectives, RMA
- Supporting Multiple Vendors and Multiple GPU Devices
**SUPPORTING GPU IN MPI COMMUNICATION (2/4)**

- **Native GPU Data Movement**
  - Multiple forms of “native” data movement
  - GPU Direct RDMA is generally achieved through Libfabrics or UCX (we work with these libraries to enable it)
  - GPU Direct IPC is integrated into MPICH

- **GPU Fallback Path**
  - GPU Direct RDMA may not be available due to system setup (e.g. library, kernel driver, etc.)
  - GPU Direct IPC might not be possible for some system configurations
  - GPU Direct (both forms) might not work for noncontiguous data
  - Datatype and Active Message Support

---

*The GPU support in MPICH is developed in close collaboration with vendor partners including AMD, Cray, Intel, Mellanox and NVIDIA*
**SUPPORTING GPU IN MPI COMMUNICATION (3/4)**

- **MPICH support for using complex noncontiguous buffers with GPU**
  - Buffer with complex datatype is not directly supported by the network library
  - Packing complex datatype from GPU into contiguous send buffer
  - Unpacking received data back into complex datatype on GPU

- **Yaksa: A high performance datatype engine**
  - Used for internal datatype representation in MPICH
  - Front-end provide interface for MPI datatypes
  - Multiple backend to leverage different hardware for datatype handle
  - Generated GPU kernels for packing/unpacking

---

The GPU support in MPICH is developed in close collaboration with vendor partners including AMD, Cray, Intel, Mellanox and NVIDIA.
- **Supporting Multiple GPU Node**
  - Data movement between GPU devices
  - Utilizing high bandwidth inter-GPU links (e.g. NVLINK)

- **GPU-IPC Communication via Active Message**
  - Create IPC handles for GPU buffers
  - Send IPC handles to target process
  - Receiver initiate Read/Write using the IPC handle

- **Fallback Path in General SHM Active Message**
  - When IPC is not available for the GPU-pair

---

The GPU support in MPICH is developed in close collaboration with vendor partners including AMD, Cray, Intel, Mellanox and NVIDIA.
GPU TRIGGERED/INITIATED COMMUNICATION

- Synchronize communication with GPU
  - Communication controlled by CPU
  - Sync cost and kernel launching cost

- MPI-4 Partitioned Communication
  - Pready/Parrived

- Stream-triggered Operation
  - Persistent request

- Passing Stream in MPI Operation
NEW COLLECTIVE INFRASTRUCTURE

- Thanks to Intel for the significant work on this infrastructure
- Two major improvements:
  - C++ Template-like structure (still written in C)
    - Allows collective algorithms to be written in template form
    - Provides “generic” top-level instantiation using point-to-point operations
    - Allows device-level machine specific optimized implementations (e.g., using triggered operations for OFI or HCOLL for UCX)
  - Several new algorithms for a number of blocking and nonblocking collectives (performance tuning still ongoing)

Contributed by Intel (with some minor help from Argonne)
SELECTING COLLECTIVE ALGORITHM

- **Choose Optimal Collective Algorithms**
  - Optimized algorithm for certain communicator size, message size
  - Optimized algorithm using HW collective support
  - Making decision on each collective call

- **Generated Decision Tree**
  - JSON file describing choosing algorithms with conditions
  - JSON file created by profiling tools
  - JSON parsed at MPI_Init time and applied to the library

*Contributed by Intel (with some minor help from Argonne)*
MPICH-4.0 ROADMAP

- MPICH-4.0a1 available at [http://github.com/pmodels/mpich](http://github.com/pmodels/mpich)
- MPICH-4.0 GA coming next this summer
  - MPI-4 support
  - Improvement on collective with GPU buffers
  - More
THANK YOU

Yanfei Guo, Assistant Computer Scientist

Argonne National Laboratory