MPICH for Exascale: Supporting MPI-4 and ECP

Yanfei Guo (ANL), Ken Raffenetti (ANL), Hui Zhou (ANL)
Agenda

• Intro

• MPICH Updates
  – GPU
  – Collective
  – MPI-4

• Partner Updates
  – Intel MPI (Nusrat Islam)
  – MVAPICH (Dr. Dhabaleswar Panda)

• Q&A
Exascale MPI (MPICH)

- Funded by DOE for 29 years
- Has been a key influencer in the adoption of MPI
  - First/most comprehensive implementation of every MPI standard
  - Allows supercomputing centers to not compromise on what features they demand from vendors
- DOE R&D100 award in 2005
- MPICH and its derivatives are the world’s most widely used MPI implementations
  - Supports all versions of MPI including the recent MPI-3.1
- MPICH Adoption in US Exascale Machines
  - Aurora, ANL, USA (MPICH)
  - Frontier, ORNL, USA (Cray MPI)
  - El Capitan, LLNL, USA (Cray MPI)
MPICH ABI Compatibility Initiative

- Binary compatibility for MPI implementations
  - Started in 2013
  - Explicit goal of maintaining ABI compatibility between multiple MPICH derivatives
  - Collaborators:
    - MPICH (since v3.1, 2013)
    - Intel MPI Library (since v5.0, 2014)
    - Cray MPT (starting v7.0, 2014)
    - MVAPICH2 (starting v2.0, 2017)
    - Parastation MPI (starting v5.1.7-1, 2017)
    - RIKEN MPI (starting v1.0, 2016)
- Open initiative: other MPI implementations are welcome to join
- [http://www.mpich.org/abi](http://www.mpich.org/abi)
MPICH Distribution Model

- Source Code Distribution
  - MPICH Website, Github
- Binary Distribution through OS Distros and Package Managers
  - Redhat, CentOS, Debian, Ubuntu, Homebrew (Mac)
- Distribution through HPC Package Managers
  - Spack, OpenHPC
- Distribution through Vendor Derivatives
MPICH Releases

• MPICH typically follows an 18-month cycle for major releases (3.x), barring some significant releases
  – Minor bug fix releases for the current stable release happen every few months
  – Preview releases for the next major release happen every few months
• Current stable release is in the 3.4.x series
  – mpich-3.4.1 was released last week
• Upcoming major release is in the 4.0 series
  – mpich-4.0a1 is released
MPICH-3.4 Series

- **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

The CH4 in MPICH is developed in close collaboration with vendor partners including including AMD, Cray, Intel, Mellanox and NVIDIA

**POC: Ken Raffenetti**
raffenet@mcs.anl.gov
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
Supporting GPU in MPI Communication (1/3)

- **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 including AMD, Cray, Intel, Mellanox and NVIDIA.

POC: Yanfei Guo
<yguo@anl.gov>
Supporting GPU in MPI Communication (2/3)

- **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 GPU in MPI Communication (3/3)

- **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
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)
Collective Selection Framework

• 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 Release Series
Implement MPI-4 Standard
and more …
Focuses in MPICH 4.0 Release Series

• Full implementation of MPI 4 specification
  – MPI Sessions
  – Partitioned Communications
  – Persistent Collectives, Tool Events, Large Count, and more
  – https://www.mpi-forum.org/docs/

• Enhance GPU and threading support
  – Make the support more stable and user experience smoother
  – Push the production performance to our research projection

• Improve useability
  – Explore MPIX space for more natural/direct semantics
MPICH 4.0 Roadmap

• MPICH-4.0a1 released in February
  – Majority of the MPI 4 API implemented

• MPICH-4.0a2 will release in June
  – Synchronized to MPI Forum meeting with the expected official ratification of MPI 4 standard
  – Full implementation of MPI 4 API
  – More stable GPU/threading support

• Beta and GA release in early 2022

• Most bug fixes will be back ported to 3.4.x
C Binding Generation

- + 3,000 lines of Python script
- - 40,000 lines of C
- API extracted from mpi-standard repo
- Generates –
  - Profiling interface
  - API documentation
  - Parameter validation
  - Handle object pointer conversion
- Fortran binding generation will be updated to Python and unified
  - F08 binding generation 80% done
MPI Session

- Libraries to keep MPI usage opaque to user

- Basic implementation internally initializes “world” in the first `MPI_Session_init/MPI_Init`

- Better implementation will delay the world initialization to first world-comm

- Fully *correct* implementation need to support first-class dynamic processes
Partitioned Communication

• In-between two-sided (pt2pt) and one-sided (RMA) communication

• Basic implementation done, plenty of optimization opportunity ahead!

```c
if (rank == sender) {
    MPI_Psend_init(buf, parts, cnt, datatype,
        recver, tag, comm, info, &req);
    MPI_Start(&req);
    ...
    MPI_Pready(i);
    ...
    MPI_Wait(&req, MPI_STATUS_IGNORE);
    MPI_Request_free(&req);
}

if (rank == recver) {
    MPI_Precv_init(buf, parts, cnt, datatype,
        sender, tag, comm, info, &req);
    MPI_Start(&req);
    ...
    MPI_Parrived(req, i, &flag);
    ...
    MPI_Wait(&req, MPI_STATUS_IGNORE);
    MPI_Request_free(&req);
}
```
Large Count API

- A large count version for every API that has a "count" or "displacement" argument (guess how many?)

- No more work-arounds!

- API use `MPI_Count`, internally we use `MPI_Aint` where-ever possible

```c
MPI_Type_contiguous_c(10000000000, MPI_INT, &my_type);

MPI_Send_c(buf, 10000000000, MPI_INT, dest, tag, comm);
```
Enhanced GPU Support

- MPICH is fully GPU-aware since 3.4.0
- But …
  - May experience degraded performance even for non-GPU app
    - GPU initialization cost, GPU pointer query cost, …
  - Cloudy with a chance of crash
    - GPU testing takes 8 hours!
  - Need GPU-aware API
- We are working on it …
  - `MPIR_CVAR_ENABLE_GPU=0` should recover full CPU-only performance
  - Fine-tuned usable GPU-testing and keeping it green
  - GPU direct IPC, GPU specific algorithm, …
- Explore MPIX extension
Better Threading Support

- Enable strong scaling with multiple VCI (virtual communication interface)
- Multi-VCI for Point-to-point implemented in 3.4.0
- Multi-VCI for RMA added in 4.0a1
- Multi-VCI for Active Messages coming

- Parallel semantics based on communicator/rank/tag
- Explore MPIX for direct threading semantics
Weekly MPICH Development Update

If you are excited with we are doing and like to get more technical –

https://bluejeans.com/266293319

Give us a whistle and we'll send you an invite.