STATUS OF OPENFABRICS INTERFACES (OFI) SUPPORT IN MPICH

Yanfei Guo, Computer Scientist

Argonne National Laboratory
OVERVIEW

- What is MPICH?
- Why OFI?
- Current Support
- 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 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
- 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
  - Hosted on Github
- Close abstraction for MPI
  - MPI community engaged from the start
- Vendor Support
  - Slingshot
  - AWS EFA
- Fully functional sockets provider
  - Prototype code on a laptop
- Strong Vendor Support
  - Intel, HPE, ParaStation, etc.
MPICH WITH CH4 DEVICE OVERVIEW

Abstract Device Interface (ADI)
- Machine-independent Collectives
- Derived Datatype Management
- Group Management

C/Fortran Bindings

Application
- MPI Interface

CH4
- Architecture-specific Collectives
- Active Message Fallback
- GPU Support Fallback
- VCI
- Stream Support

Netmods
- libfabric
- UCX

Shmmod
- POSIX
- XPMEM
- GPU IPC

Yaksa Datatype Engine

MPL (Portable Functionalities)
- CUDA
- HIP
- OneAPI

HWLOC

JSON-C

© OpenFabrics Alliance
Support for MPI 4.1 Specification
- mpi_memory_alloc_kinds info hint
- MPI_Request_get_status_{all,any,some}
- MPI_Remove_error_{class,code,string}
- MPI_{Comm,Session}_{attach,detach}_buffer
- MPI_BUFFER_AUTOMATIC
- Split type MPI_COMM_TYPERESOURCE_GUIDED

New Experimental Features
- MPIX Thread Communicator
- MPI-5 ABI

Enhanced GPU (esp. ZE) Support
MPI THREAD COMMUNICATOR

- MPI × Threads paradigm
  - Easy migration from MPI-only to MPI+OpenMP
- Internal Mechanism
  - On-node threads: send/recv ~ ld/st
  - Off-node threads: mapping to different communication contexts
- Supported Ops
  - Pt2pt, blocking collectives

```c
#include <mpi.h>
#include <stdio.h>
#include <assert.h>

#define NT 4

int main(void) {
  MPI_Comm threadcomm;
  MPI_Init(NULL, NULL);
  MPI_Threadcomm_init(MPI_COMM_WORLD, NT, &threadcomm);

#pragma omp parallel num_threads(NT)
{
  assert(omp_get_num_threads() == NT);
  int rank, size;
  MPI_Threadcomm_start(threadcomm);
  MPI_Comm_size(threadcomm, &size);
  MPI_Comm_rank(threadcomm, &rank);
  printf(" Rank %d / %d\n", rank, size);
  // MPI operations over threadcomm */
  MPI_Threadcomm_finish(threadcomm);
}

MPI_Threadcomm_free(&threadcomm);
MPI_Finalize();
return 0;
}
```

$ mpicc -openmp -o t.t.c
$ mpirun -n 2 ./t

<table>
<thead>
<tr>
<th>Rank</th>
<th>4 / 8</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Rank</td>
<td>7 / 8</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Rank</td>
<td>5 / 8</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Rank</td>
<td>6 / 8</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Rank</td>
<td>0 / 8</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Rank</td>
<td>1 / 8</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Rank</td>
<td>2 / 8</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Rank</td>
<td>3 / 8</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
LATENCY AND BANDWIDTH

- Only practical difference
- No fundamental difference
- See paper for detailed discussions

```
#pragma omp parallel
{
    MPI_Threadcomm_start(comm);
    #ifdef USE_MPI
        MPI_Barrier(comm)
    #else
        #pragma omp barrier
    #endif
    MPI_Threadcomm_finish(comm);
}
```
int sum[N];
#ifdef USE_MPI
    #pragma omp parallel
    {
        MPI_Threadcomm_start(comm);
        int my[N];
        int tid = omp_get_thread_num();
        for (int i = 0; i < N; i++) my[i] = tid;
        MPI_Reduce(my, sum, N, MPI_INT, MPI_SUM, 0, comm);
        MPI_Threadcomm_finish(comm);
    }
#endif
    #pragma omp parallel reduction(+:sum[:N])
    {
        int tid = omp_get_thread_num();
        for (int i = 0; i < N; i++) sum[i] = tid;
    }
#endif
PETSC + THREADCOMM PERFORMANCE

PETSc SpMV Performance (GFlops/sec)

PETSc KSP solve Performance (x 10^6 DOFs/sec)
SUPPORT MPI-5 ABI

- A working proposal currently being developed in MPI Forum
- Build once, work with either MPICH or Open MPI derivatives
- MPICH-4.2 support both MPICH ABI and optionally MPI-5 ABI
  - mpicc builds MPICH ABI, mpicc_abi builds MPI-5 ABI
  - libmpi.so implements MPICH ABI, libmpi_abi.so implements MPI-5 ABI
  - mpi.h will effectively become mpi_abi.h when mpicc_abi is used. User code always #include <mpi.h>
- MPICH-4.3a1 2H 2024
- MPICH-4.3b1 targeted for SC24
  - 4.2.x branch will be created
- GA release in early 2025
- Critical bug fixes will be backported to 4.2.x
MPICH 4.3 FEATURES

- **Standard and Quality of Life Improvements**
  - Enhance support for MPI sessions
  - Support mpi_memory_alloc_kinds side document specifications
  - Support runtime loading of selected dependency libraries (e.g. libfabric, CUDA, ROCm, etc.)
  - Continue prototyping standard MPI ABI

- **Performance Optimization and New Architectures**
  - Optimized partitioned communication
  - Support dynamic VCIs
  - Performance improvement Yaksa Datatype Engine
  - Collective arch overhaul for better support of topology aware collective algorithms and external CCL libraries (libfabric collectives, UCC, etc.)
MPI COLLECTIVE WITH LOSSY COMPRESSION

- Integrating Lossy Compression with MPI Collective for Large Message Transfer
- Efficient Scheduling of Compression and Communication
- Relying on Regular MPI P2P

Jiajun Huang, et al. gZCCL: Compression-Accelerated Collective Communication Framework for GPU Clusters, accepted by ICS 2024
THANK YOU
Yanfei Guo, Computer Scientist
Argonne National Laboratory