Perspective and Experience with OFI in MPICH

Ken Raffenetti
Software Development Specialist
Argonne National Laboratory
Email: raffenet@mcs.anl.gov
Web: http://www.mcs.anl.gov/~raffenet
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 (including MPI-1, MPI-2.0, MPI-2.1, MPI-2.2, and MPI-3.0)
- Active development lead by Argonne National Laboratory and University of Illinois at Urbana-Champaign
  - Several close collaborators who contribute many features, bug fixes, testing for quality assurance, etc.
    - IBM, Microsoft, Cray, Intel, Ohio State University, Queen’s University, Mellanox, RIKEN AICS and many others
- Current stable release is MPICH-3.1.4
- www.mpich.org
MPICH: Goals 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”
Motivations

- Why are we interested in OFI?
  - Not limited to a single hardware configuration
  - Actively, openly developed
  - OFI provides a nice abstraction for MPI
    - Less code
    - Hides nitty-gritty details
  - Promise of a fully functional sockets provider for laptop development
MPICH Layered Design

- MPI
- MPICH
- ADI3
- CH3

Channel Interface
- Nemesis

Netmod Interface
- TCP
- MXM
- IB
- Portals4
- OFI
Why a CH3/Nemesis Netmod?

- Provides MPI correctness (all of MPI-3)
  - Years of testing and bugfixes
- Highly-tuned shared memory transport
- Netmod supports hardware matching
- Upcoming improvements in MPICH 3.2 release series
  - RMA scalability improvements
  - New netmod hooks
OFI Netmod

- Network Initialization
  - Address discovery/exchange

- Data movement
  - Send/Recv

- Control messages
  - Also involves data movement
OFI Netmod

- **Initialization**
  - Provider selection
    - Tag matching (FI_TAGGED)
    - Dynamic memory region spanning all memory (FI_DYNAMIC_MR)
  - Endpoint creation
    - Reliable Datagram
    - Address exchange over PMI to populate AV
      - Stored in MPICH virtual connection table
OFI Netmod

- Point-to-point data movement
  - Closely maps to fi_tsend/trecv functionality

```c
MPI_Send(buf, count, datatype, dest, tag, comm)
fi_tsend(gl_data.endpoint,              /* Endpoint */
        send_buffer,                       /* Packed or user */
        data_sz,                               /* Size of the send */
        gl_data.mr,                         /* Dynamic memory region */
        VC_OFI(vc)->direct_addr, /* VC address */
        match_bits,                        /* Match bits */
        &(REQ_OFI(sreq)->ofi_context));
```
Pt2Pt Benchmarks (Blues cluster @ ANL)

- 0-byte PingPong
  - 1.90 μs MPICH/OFI vs 1.44 μs MVAPICH/PSM
OFI Netmod

- Control Messages and RMA
  - MPICH CH3 implementation based on active messages
    - Use of persistent request to accept incoming CH3 packets + eager data
    - Received into temporary buffer, then copied to user buffer
    - Ongoing work in MPICH 3.2 to provide put/get overrides in netmod
Put Benchmarks

- **1-byte Put Latency**
  - 11.52 μs (MPICH/OFI) vs. 8.92 μs (MVAPICH/PSM)

- **Bandwidth**
OFI/Portals 4 Comparison

- **Similarities**
  - Shared, connection-less endpoints
  - Both one-sided and two-sided primitives
  - Hardware matching
  - Network hardware independent

- **Differences**
  - Queue management
    - Portals 4 – explicit unexpected queue management
    - OFI – single persistent request
  - Flow-control
    - Portals 4 - leaves recovery to the upper layer
    - OFI – enabled or disabled

Additional Complexity
Future Work

- How can we improve our OFI support?
  - Finish CH3/Nemesis improvements
  - Support providers with different sets of functionality?
  - Triggered operations?
Thank you

- Questions?