Towards Architecture-aware Hierarchical Communication Trees on Modern HPC Systems

Presentation at the 10th Annual MVAPICH User Group (MUG) Meeting (MUG '22)

Bharath Ramesh, Jahanzeb Maqbool Hashmi, Shulei Xu, Aamir Shafi, Mahdieh Ghazimirsaeed, Mohammadreza Bayatpour, Hari Subramoni, and Dhabaleswar K. Panda

The Ohio State University

ramesh.113@osu.edu

Follow us on

https://twitter.com/mvapich

Presentation Outline

- Introduction/Background
- Benchmark-level results for MPI_Allreduce and MPI_Bcast
- Application-level results
- Conclusion/Summary

Overview

- Modern multi-cores machines are complex
  - Hundreds of cores
  - Multiple levels of memory hierarchies
- Algorithms are based on abstract notion of the systems
  - E.g., An MPI rank participating in a broadcast
- Performance depends on software-level designs
  - How to map abstract notions used by algorithms on to real systems?
- Systems are changing and implementations are not able to cope-up
  - What happens to a socket-aware algorithm on a system with multi-NUMA and multi-CCX per NUMA e.g., ROME?

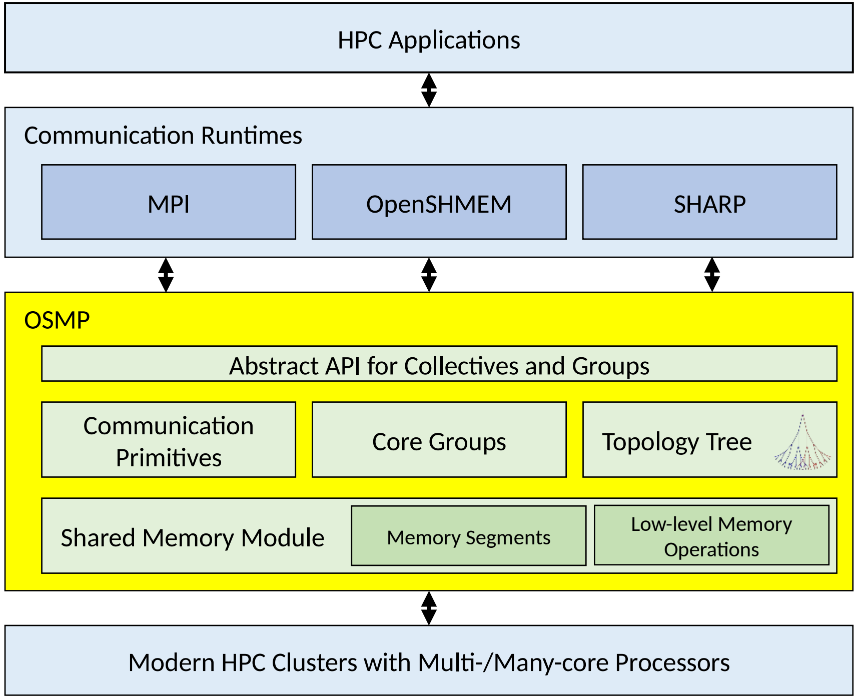
Motivation

- Problem-1
  - Implementations try to emulate the algorithms without notion of the hardware
  - Example: Processes (ranks) doing send/recv to replay a reduce-scatter allgather or recursive doubling Allreduce
- Problem-2
  - Implementations try to cater to hardware but not robust enough
  - Example: a socket-aware algorithm to minimize cross-socket traffic
- Observations
  - Modern systems have multiple levels of memories
    - Multiple coherence domains
  - Hardware should be the first-class citizen
    - Implementation should emulate the Hardware instead of abstract algorithms
    - High-level concepts should be translated for a given hardware first

OSMP (Optimized Shared Memory Processing)

- OSMP is an attempt to design shared memory communication on modern systems
- Design principle
  - “*Hardware as a first-class citizen*”
  - High-level algorithms should be *machine-aware*
- Key ideas
  - Software abstractions for hardware features e.g., topologies, distances, core-to-core communications
  - Software resources are placed on to coherence domains for locality.
  - Each hardware coherence domain has software resources e.g., synchronization flags, send/recv queues, etc.
  - Memories are local to coherence domains and tied up with hardware abstractions.

OSMP high level architecture

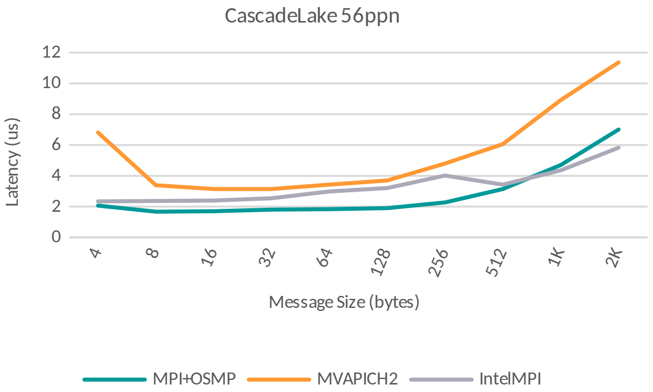


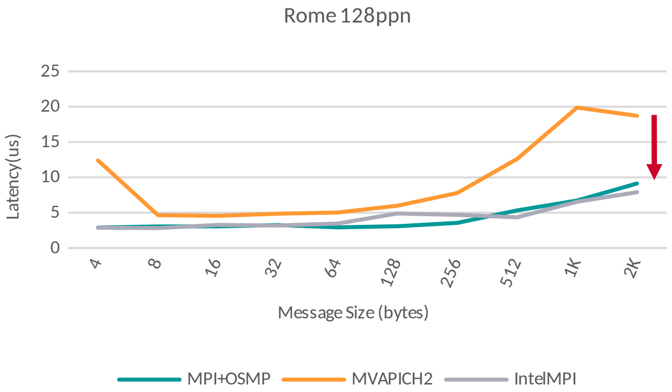
- The OSMP framework can be used by different communication runtimes
- Contains APIs for collective operations, abstractions for trees and other operations
- User implements high-level algorithms
  - Some are provided by default as well
- OSMP applies the high-level algorithms to internal abstractions

Implementing algorithms with OSMP using barrier as an example

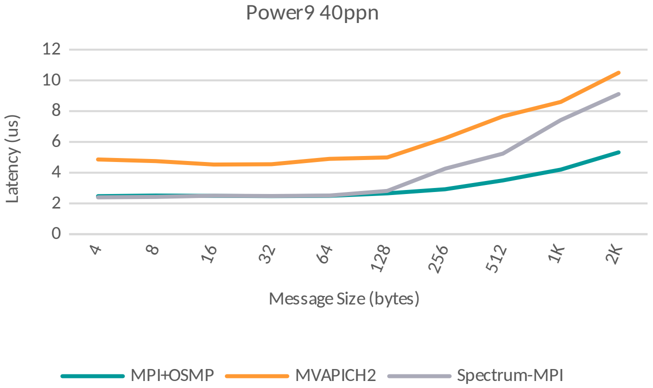
- User implements "base-case" arrival and notify routines (assuming a group of processes)
  - Arrival used to tell the root process that every other process has reached the barrier
  - Notify used by the root process to tell every other process that it has received the "arrived" message
- OSMP takes the "base-case" implementation and runs it on every level of the abstracted tree
  - Essentially converts the "base-case" algorithm to a topology aware version by walking and executing it on a tree
  - Trees used by OSMP can be a regular hierarchy (NUMA, socket, system etc.) or "virtual"

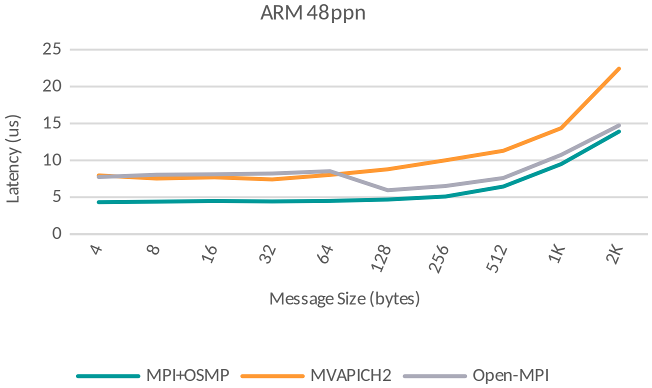
Micro-benchmark results for MPI_Allreduce



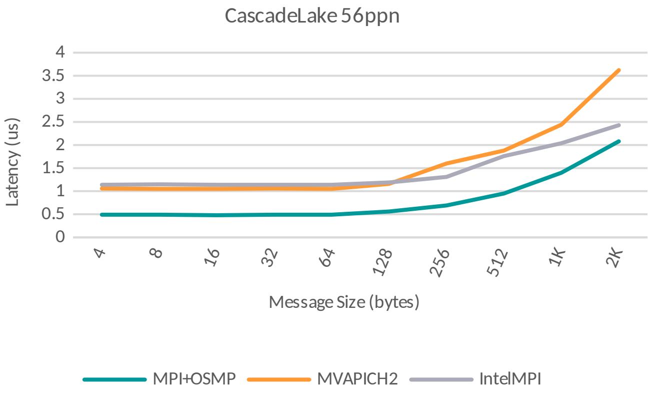


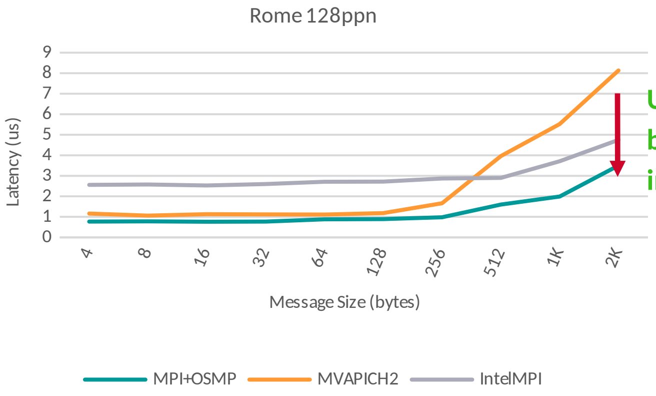
Up to 2X over base-case implementation



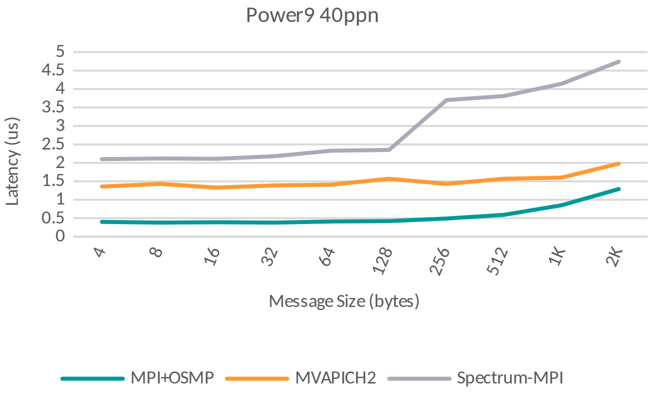


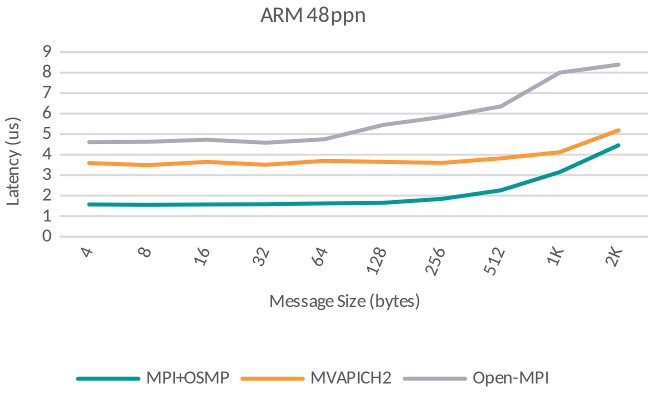
Micro-benchmark results for MPI_Bcast



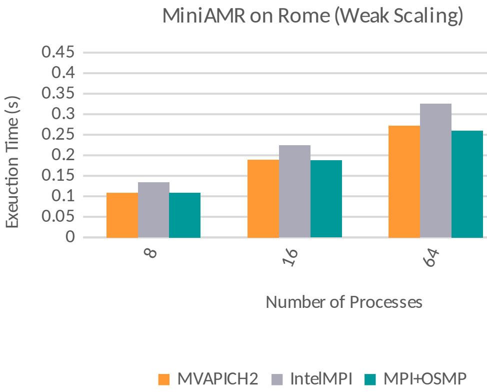


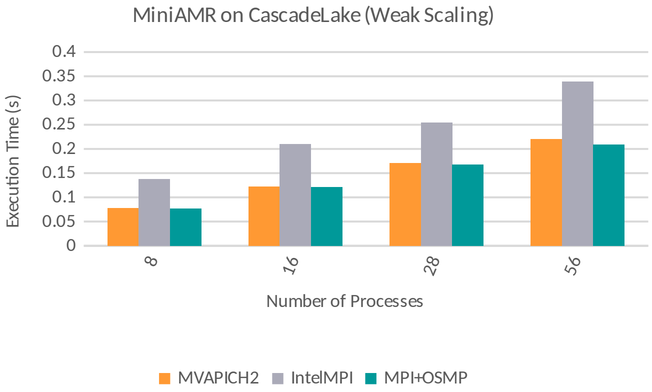
Up to 2.3X over base-case implementation

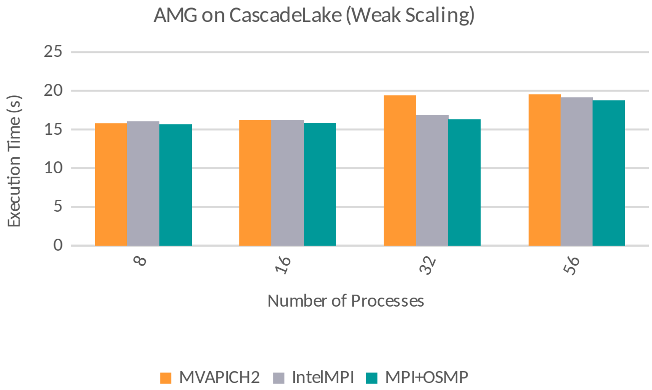




Application-level results







Up to 18.8% improvements for 32 processes over base-case implementation for AMG on CascadeLake

Conclusion and Future work

- Conclusion
  - Having simple algorithms are effective when the right abstractions are used
  - Designs extendable to future architectures by changing the topology trees – No change required with respect to algorithms
- Future work
  - Explore abstractions for other inter-process transfer mechanisms (such as XPMEM)
  - Add support for point to point and large message collectives
  - Extend topology and abstractions beyond the node (Example : Network Topologies)

Thank You!

ramesh.113@osu.edu

Network Based Computing Laboratory

Follow us on

https://twitter.com/mvapich

Network-Based Computing Laboratory

http://nowlab.cse.ohio-state.edu/

MVAPICH

MPI, PGAS and Hybrid MPI+PGAS Library

The High-Performance MPI/PGAS Project

http://mvapich.cse.ohio-state.edu/

HiBD

High-Performance Big Data

The High-Performance Big Data Project

http://hibd.cse.ohio-state.edu/

HiDL

High-Performance Deep Learning

The High-Performance Deep Learning Project

http://hidl.cse.ohio-state.edu/