MVAPICH
MPI, PGAS and Hybrid MPI+PGAS Library

Optimizing Communication Performance of Derived Datatypes

9th MVAPICH User Group (MUG) Conference '21

Presentation at MUG '21

Kaushik Kandadi Suresh

Network Based Computing Laboratory (NBCL)

Dept. of Computer Science and Engineering, The Ohio State University

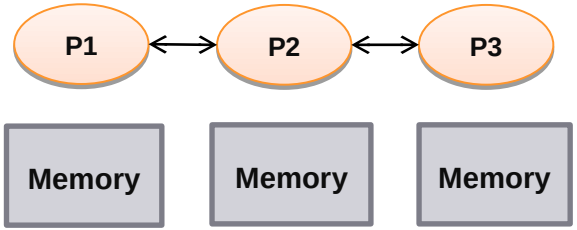
kandadisuresh.1@osu.edu

Outline

- Introduction
- Background – MPI Derived Datatypes
- Problem Statement
- Performance evaluation
- Conclusion & Future works

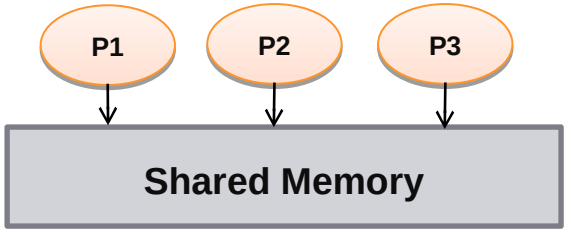
Introduction to MPI

- Programming models provide abstract machine models
- Models can be mapped on different types of systems
  - e.g. Distributed Shared Memory (DSM), MPI within a node, etc.
- Programming models offer various communication primitives
  - Point-to-point (between pair of processes/threads)
  - Remote Memory Access (directly access memory of another process)
  - Collectives (group communication)
- MPI provides datatypes for exchanging messages
- Intrinsic types
  - MPI_INT, MPI_DOUBLE, etc.
- Derived Datatypes (DDT)
  - MPI_Type_Contiguous, MPI_Type_Vector, MPI_Type_Indexed, etc.



Distributed Memory Model

MPI (Message Passing Interface)

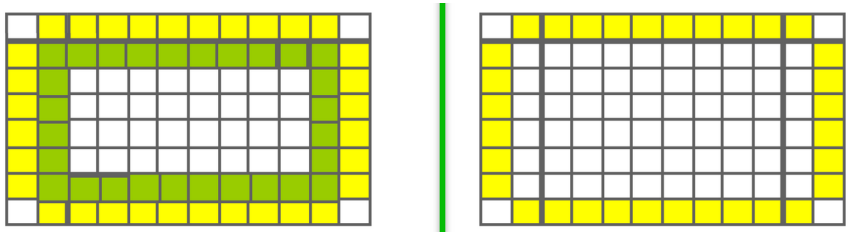


Shared Memory Model

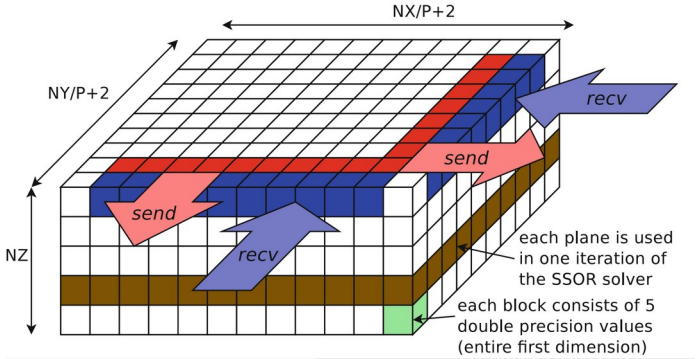
SHMEM, DSM

Existence of Non-Contiguous Memory Layouts in HPC

- 2D Stencil Data Layout



- Data Layout in NAS LU

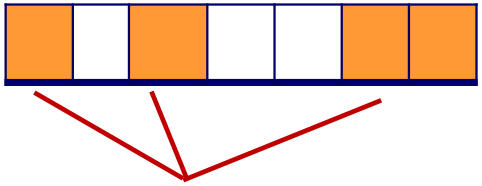


1) https://www.mcs.anl.gov/~thakur/sc16-mpi-tutorial/slides.pdf
2) Schneider T., Gerstenberger R., Hoefler T. (2012) Micro-applications for Communication Data Access Patterns and MPI Datatypes. In: Träff J.L., Benkner S., Dongarra J.J. (eds) Recent Advances in the Message Passing Interface. EuroMPI

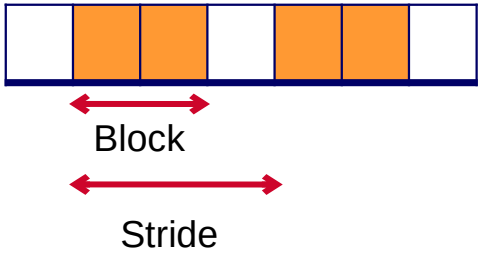
Types MPI Derived Datatypes

MPI_Type_contiguous

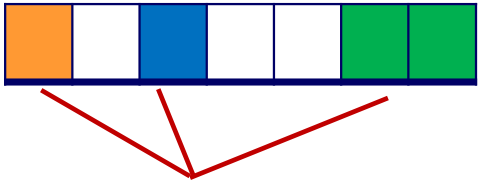
MPI_Type_indexed



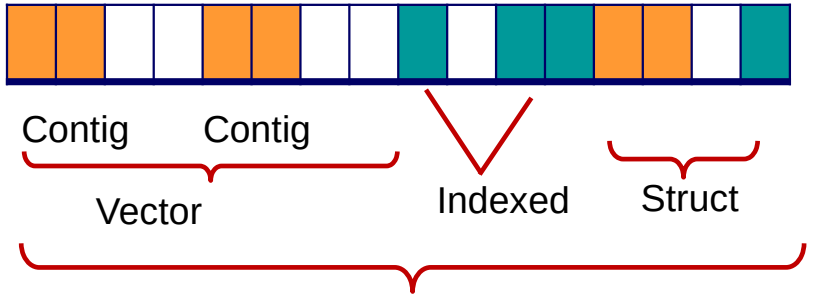
MPI_Type_vector



MPI_Type_struct



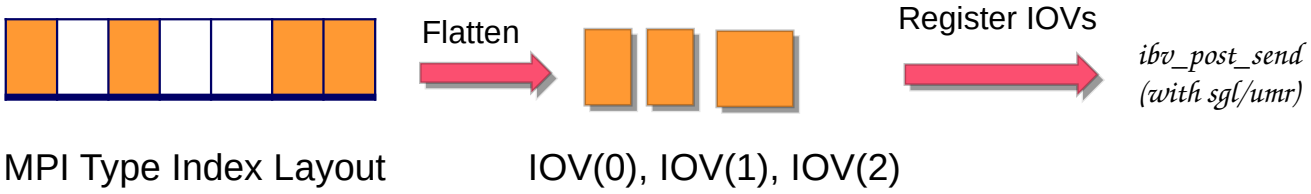
Nested Type Example:
(Combination of type_contiguous, type_indexed, type_vector, type_struct)



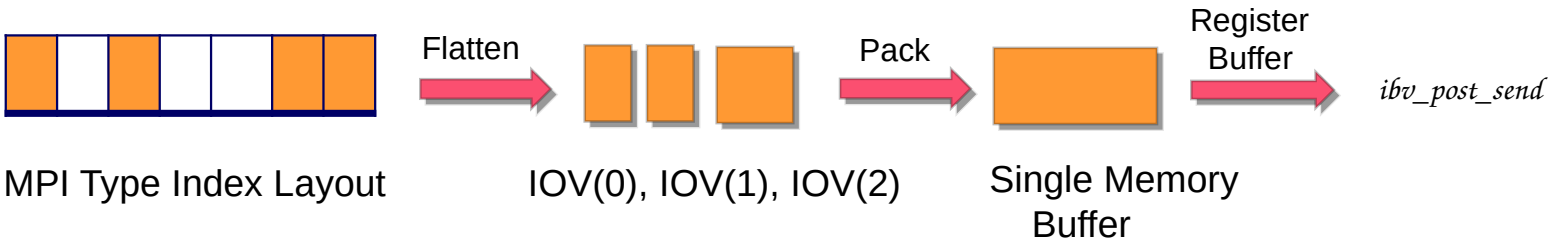
Courtesy: https://www.mcs.anl.gov/~thakur/sc16-mpi-tutorial/slides.pdf

Types of MPI DDT Transfer

(a) Hardware Assisted : Uses SGL/UMR based transfer

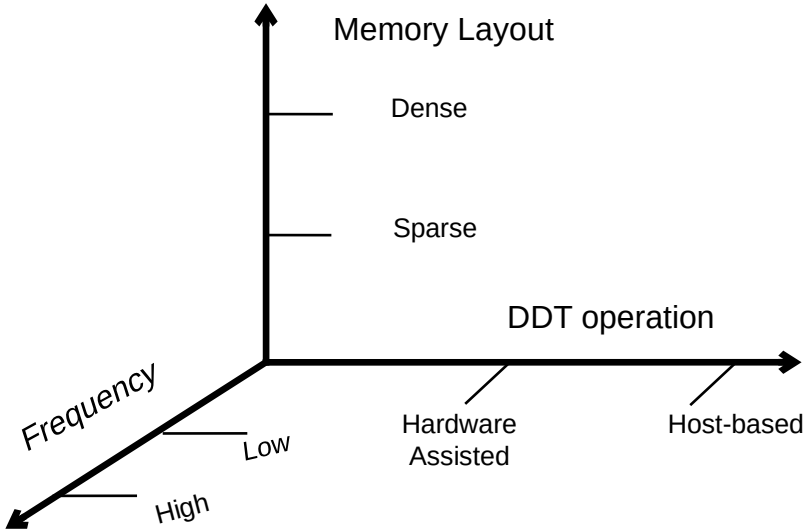


(b) Host based : Uses CPU to pack/unpack



The missing Big Picture!

- Performance of DDT depends on different factors:
  - DDT operation used [Pack/Unpack, Hardware Assisted]
  - DDT scheme [Sequential, Pipelined]
  - Layout reuse frequency [Low, High]

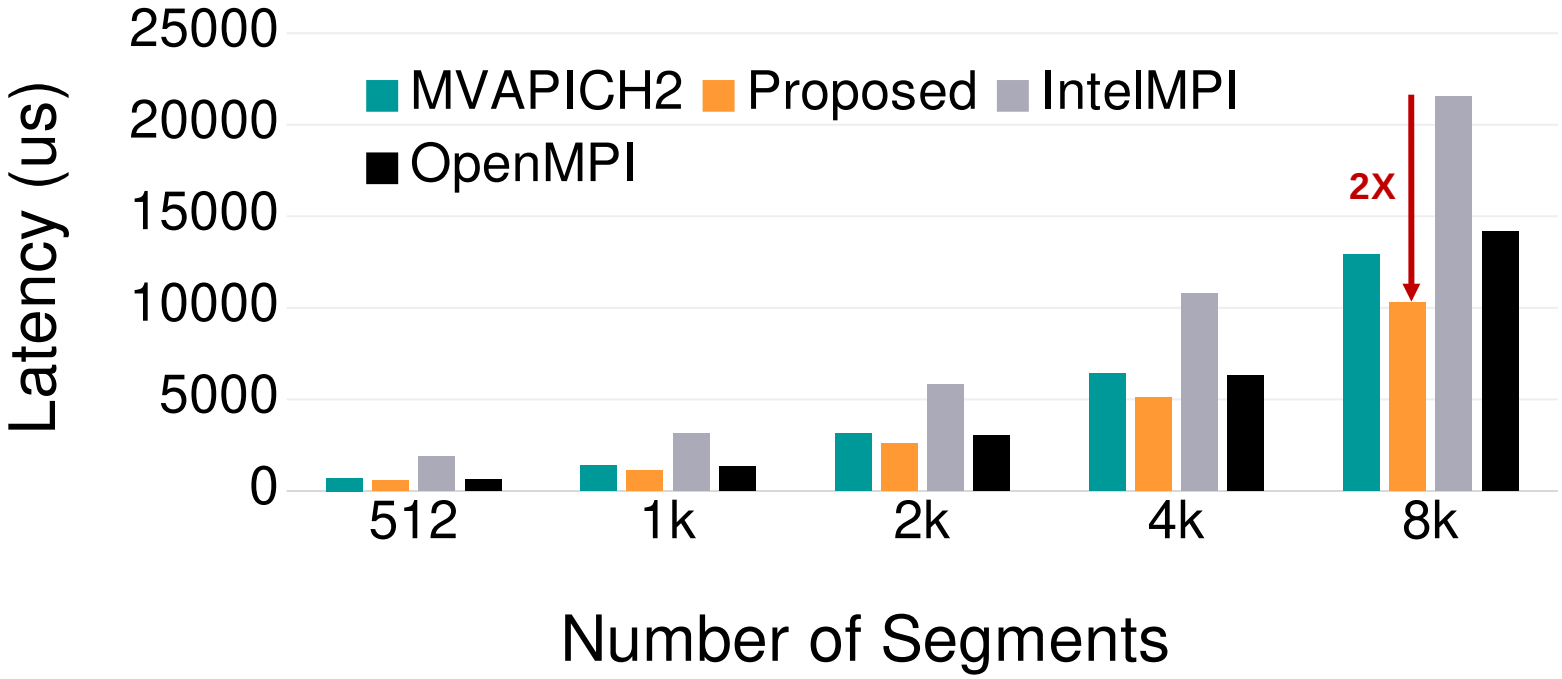


Experimental Setup

Processor Family	Intel Xeon
Processor Model	Platinum 8280
Clock Speed	2.7 GHZ
No. of Sockets	2
Cores per Socket	28
RAM	192 GB
Interconnect	Mellanox HDR-100 ConnectX-6
MPI Library	MVAPICH2-X 2.3rc2, OpenMPI 4.1, IntelMPI 2019
Benchmarks and Application	1) OSU Microbenchmark (OMB) with Vector DDT 2) DDT bench 3) MiniGhost

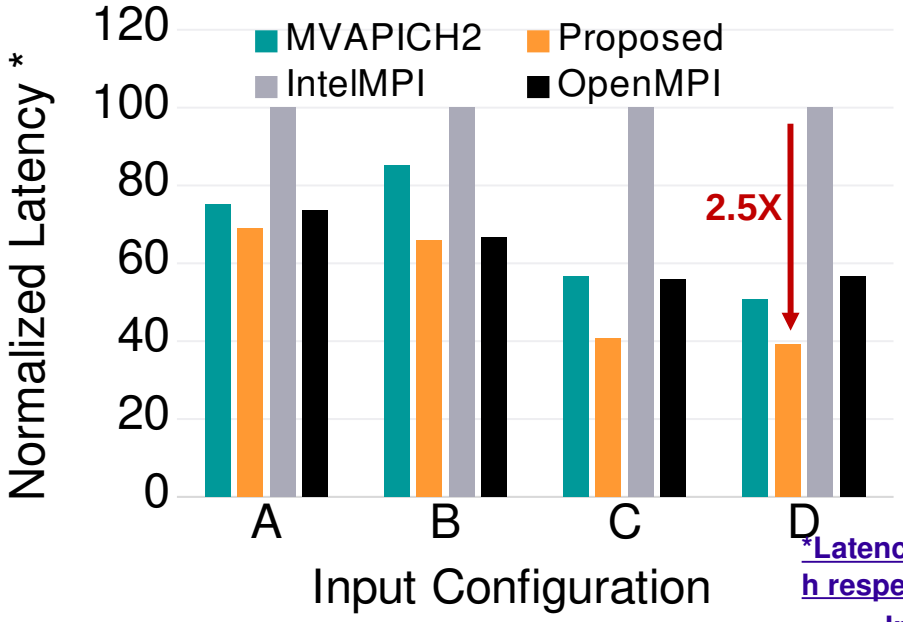
Performance of vector benchmark

- Vector of Block Length – 4KB
- Improvement up-to 30% over OpenMPI and 2X over IntelMPI

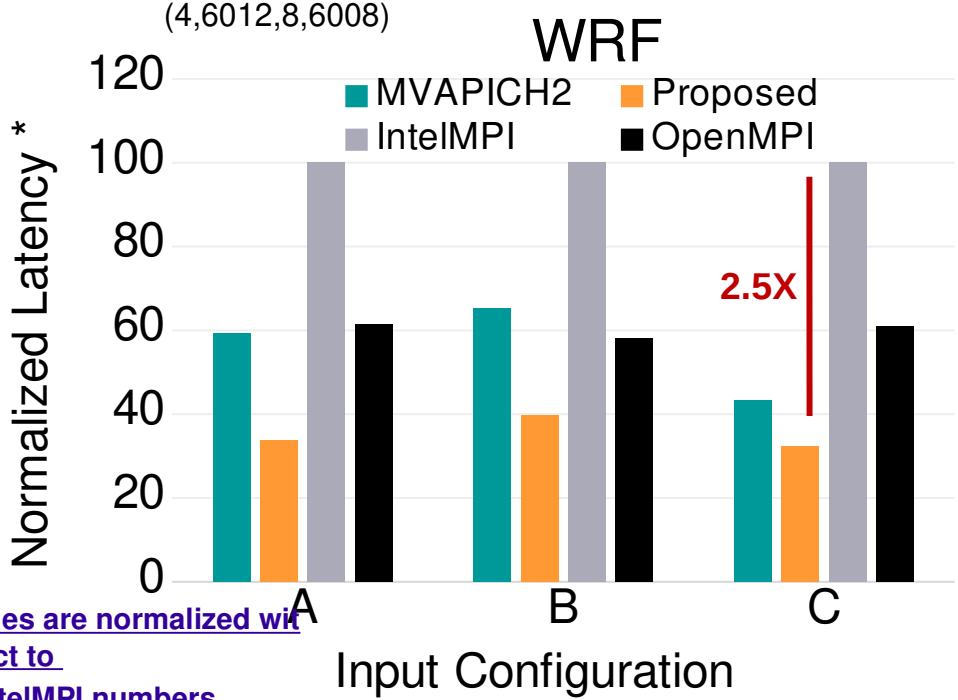


Performance of DDTbench

- NASMG: Block length is 8 bytes for X-direction and 256 bytes to 5KB in the Y-direction
- 28% improvement over MVAPICH2 and 2.5X over IntelMPI
- Inputs : A = (256,32,32) B = (512,66,66) C = (2048,66,120) D = (512,32,24)

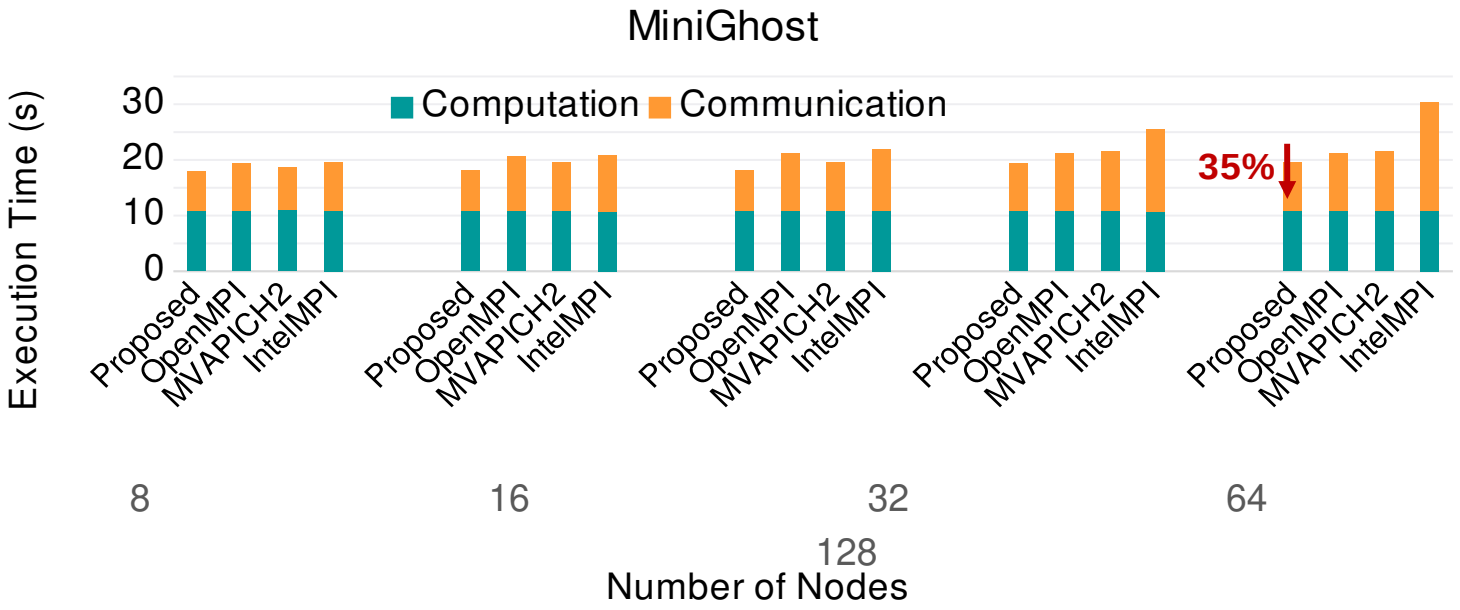


- WRF: The datatypes used in WRF are struct of vectors for both X and Y direction
- We see improvements up to 1.75X compared to MVAPICH2 and up to 2.5X improvements over IntelMPI
- Inputs : A = (4,4018,8,4010) B = (4,2060,8,2056) C = (4,6012,8,6008)



*Latencies are normalized with respect to IntelMPI numbers

Performance of MiniGhost application



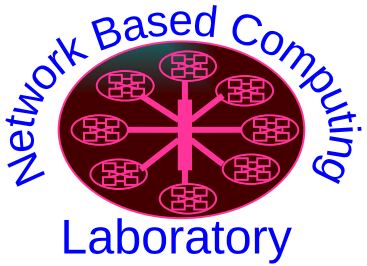
- Execution time of the proposed scheme is up to 35% better than Intel-MPI, 7.8% better than OpenMPI, and 9% better than MVAPICH2 at a scale of 128 nodes.

Conclusion and Future work

- Conclusion
  - DDT cost is impacted by transfer schemes, memory layouts, and DDT operation
  - Proposed dynamic scheme that considers:
    - Memory Layout
    - Frequency
    - DDT operation
  - Proposed design achieves up to 3X improvement in performance over state-of-the-art MPI libraries at the micro-benchmark level.
  - Demonstrated up to 35% improvement in MiniGhost performance at 128 nodes
- Future work
  - Comprehensive evaluation at large scales for more HPC applications
  - Scaling studies with larger number of processes per node

Thank You!

Kandadisuresh.1@osu.edu



Network-Based Computing Laboratory
http://nowlab.cse.ohio-state.edu/

Follow us on

https://twitter.com/mvapich

The MVAPICH2 Project
http://mvapich.cse.ohio-state.edu/