MVAPICH

MPI, PGAS and Hybrid MPI+PGAS Library

HiBD

High-Performance Big Data

HiDL

High-Performance Deep Learning

Towards Dynamic Message Passing Protocols For Stencil-based Communication Patterns

Kaushik Kandadi Suresh, Bharath Ramesh, **Goutham Kalikrishna Reddy Kuncham**,

Hari Subramoni, Dhabaleswar K. (DK) Panda

IEEE Cluster 2025

08/20/2025

Network-based Computing Laboratory

Department of Computer Science and Engineering

The Ohio State University

Table of Contents

Introduction – Problem Space

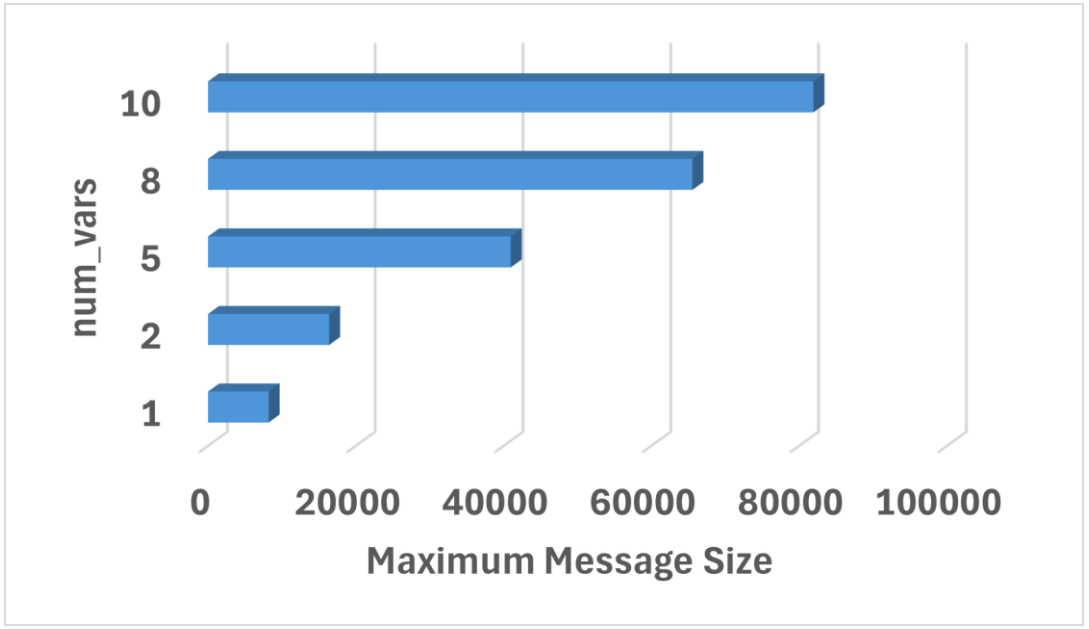
- Halo-exchange communication patterns occur in many stencil-based HPC applications such as MiniAMR, MiniGhost, and MILC.
- In this pattern, each process often performs a mix of inter-node (between different compute nodes) and intra-node (within the same compute node) transfers.
- Depending on the input and processor grid size, the amount of time spent in inter-node or intra-node communication could dominate the total communication time.

The challenge: Efficiently managing these diverse communication patterns is challenging, especially given the dynamic nature of real-world applications and the limitations of current MPI libraries

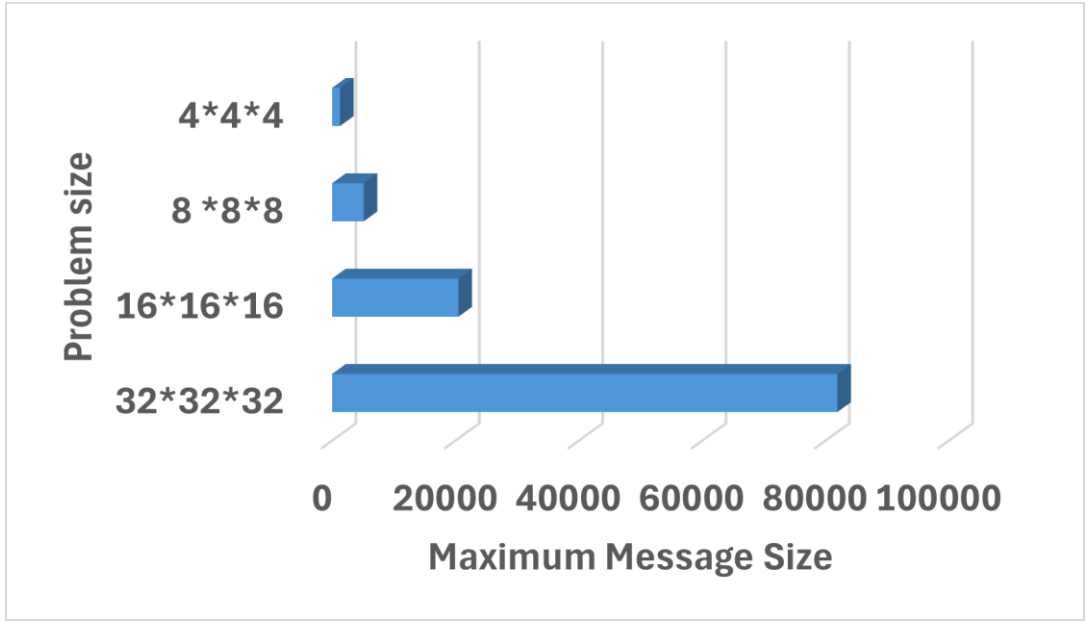
Our goal: To propose dynamic message passing protocols that significantly optimize communication time for these patterns

Table of Contents

Motivation 1: Changing Message Patterns in Applications



Fixed grid size in miniAMR (32x32x32)



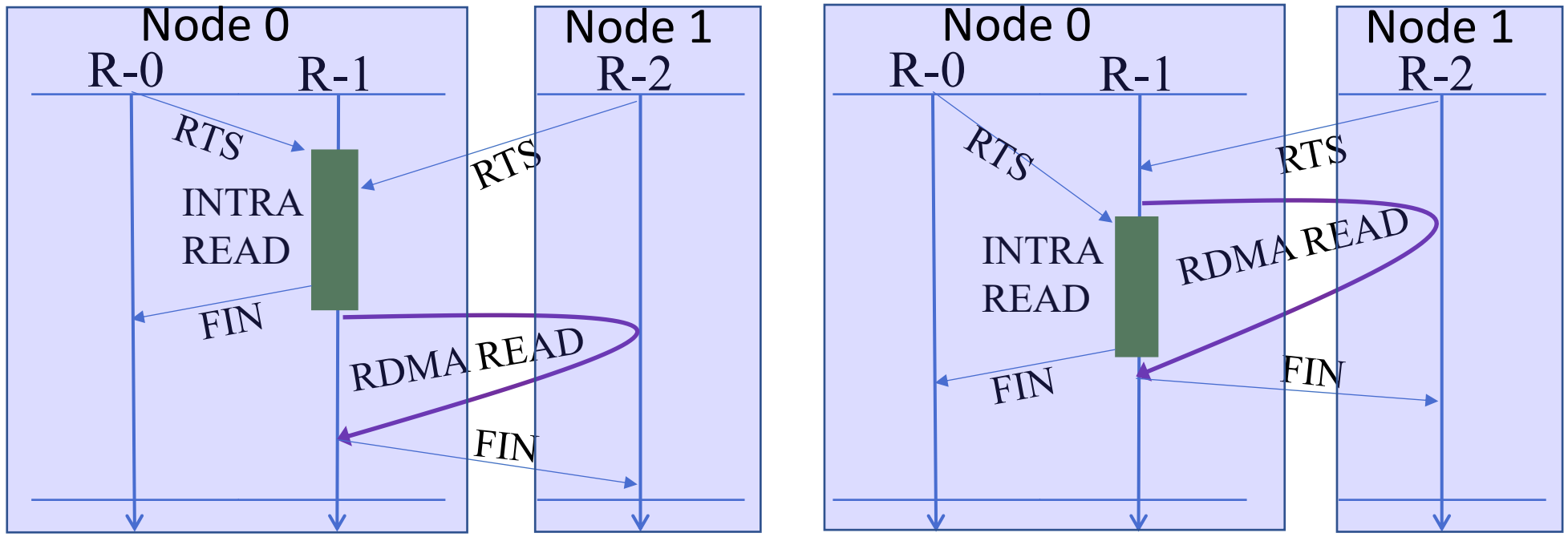
Fixed num_vars in miniAMR (10)

- Application parameters (e.g., problem grid dimensions, processor grid dimensions, number of variables) significantly affect message sizes
- For example, in MiniAMR, the average message size can vary drastically just by changing the number of variables for the same problem size

Motivation 1: Lack of Adaptivity in Intra-Node Communication

- Existing work-stealing solutions for intra-node communication often use a static, on-demand chunking scheme
  - This means a fixed number of chunks for every message size
  - Susceptible to skews in the application. The same chunk size does not necessarily work everywhere
  - Impractical to tune - Static tuning for each specific message size and CPU architecture
- Fixed protocols for a given message size
  - Work stealing has additional overheads, so switching to a basic protocol might be better if work stealing is unnecessary
- Can we design protocols that adapt to application patterns and dynamically select chunk sizes/protocols at runtime?

Motivation 2: Problem with Rendezvous Protocol



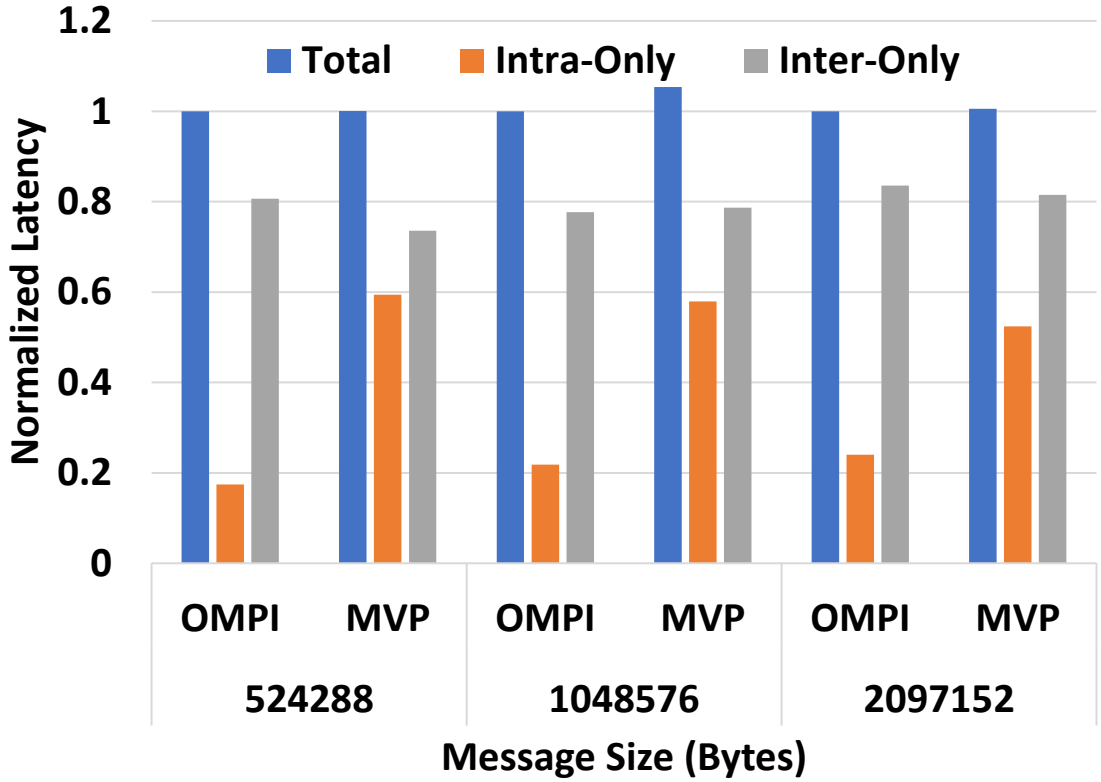
(a) Missed Overlap Opportunity

(b) Ideal Overlap

- Current MPI libraries often fail to achieve optimal overlap between intra-node and inter-node transfers
- Rendezvous protocol, commonly used for large messages, requires a handshaking mechanism (Request-to-Send (RTS) and Clear-to-Send (CTS)) before data transfer
- The rendezvous protocol does not guarantee overlap
- It depends on the order in which control message arrives
- Can we design a protocol to improve the overlap potential without additional copy for stencil workloads?

Example: Lack of Overlap in Communication Runtimes

- 7-point stencil communication benchmark
  - Up-to 6 neighbors (2 per direction)
- Comparison of OpenMPI, MVAPICH-Plus on 4 Nodes, 144 PPN
  - Total – send/recv to/from all neighbors
  - Inter-only – sends/recvs to/from inter-node neighbors only
  - Intra-only – sends/recvs to/from intra-node neighbors only
- Ideal latency = max(inter-only, intra-only)
  - inter-only >> intra-node
  - Ideal latency = inter-only
- Observation: total latency = 1.3X inter-only latency



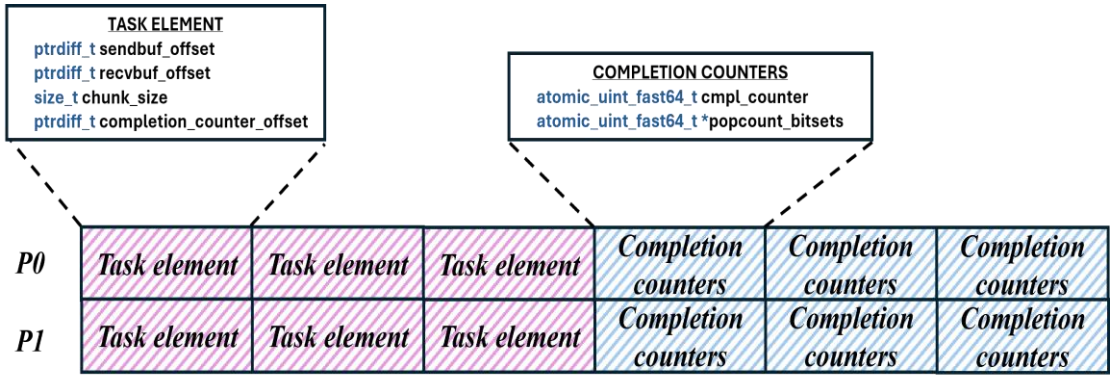
7-Point Stencil Communication Benchmark

4 Nodes 144 PPN

Table of Contents

Queue designs

- Two queue types
- SPSC queues for metadata/control msgs
  - Single producer, single consumer
  - N^2 queues for N processes
  - Use release store and acquire load + caching to improve performance
- SPMC queues for work-stealing
  - Single producer, multiple consumer
  - N queues for N processes
  - The producer keeps writing to the queue. Consumers need to catch up.
- Bitmask to track the number of work-stealers available



Dynamic stealing protocol (Receiver-initiated)

- Sender
  - Enqueue ready to send (RTS) to receiver in SPSC queue
  - Wait for CTS (if cooperative) or FIN (if GET/work stealing)
- Receiver
  - Split the buffer into a small pre-defined chunk size (say, 1024 bytes) in the first iteration
  - Subsequent iterations use set completion bits (popcount) to determine number of chunks
  - If popcount is 1, use a GET protocol and send FIN. If it's 2, use COOP -> send CTS and do memcpy. Otherwise, use the generic work-stealing protocol.
- Progress
  - Poll self SPMC queue and dequeue task first and then poll remote SPMC queue if nothing
  - Handle work stealing memory copy (if any), and perform an atomic fetch or on bitset
  - On request completion (during test/wait), use popcount to update number of stealers for a given peer, and then send a FIN message.

Designing the Overlap Protocol

- Key Idea: Instead of the application providing the receive buffer, the communication library itself manages the receive buffers.
- We observed that all stencil-based applications use auxiliary buffers for communication
- We propose simple extensions to the MPI standard to allow the library to manage buffers.
- We propose to modify the MPI_Irecv semantics to allow a NULL value for the receive buffer, indicating that the library will provide the receive buffer when a NULL value is given to MPI_Irecv.

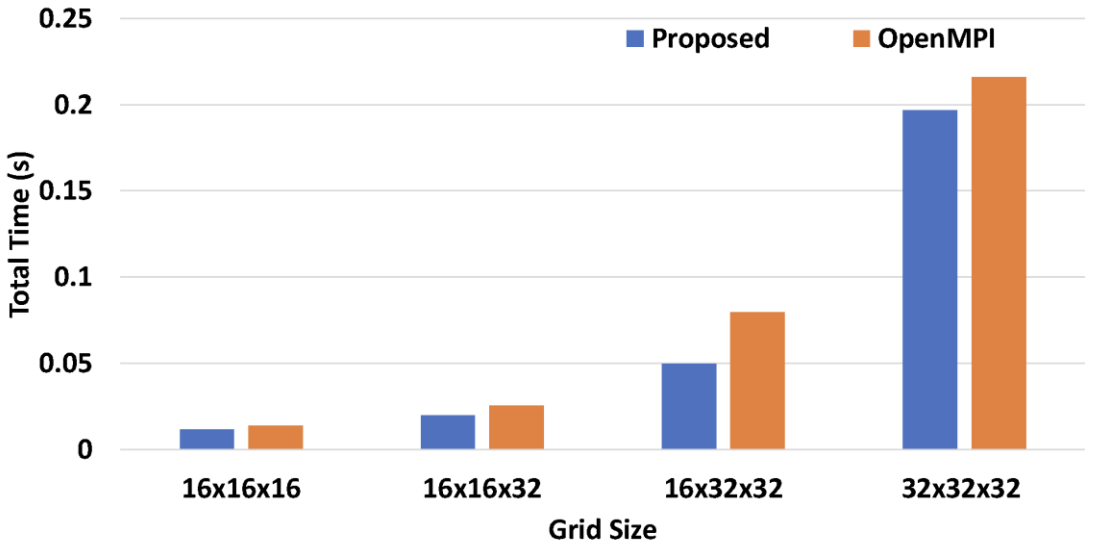
Proposed Modifications to MPI Standard

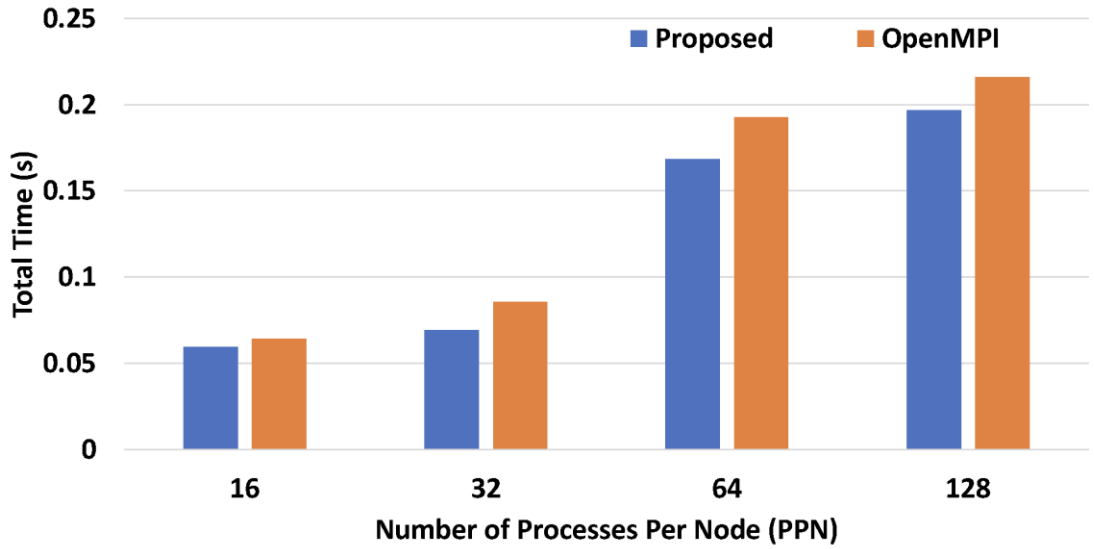
```
1  /* Get the Receive Buffer After Completion */
2  int MPIX_Get_recv_buf(MPI_Status status,
3                        void **recv_buf);
4  /* Release the Receive Buffer to the MPI library. */
5  int MPIX_Release_recv_buf(MPI_Status status,
6                        void *recv_buf);
```

- Protocol managed receive buffer
- Exposed via MPIX_Get_recv_buf method
- Release the buffer for re-use using MPIX_Release_recv_buf
- Why this works:
  - Stencil based applications use auxiliary buffers for communication

Table of Contents

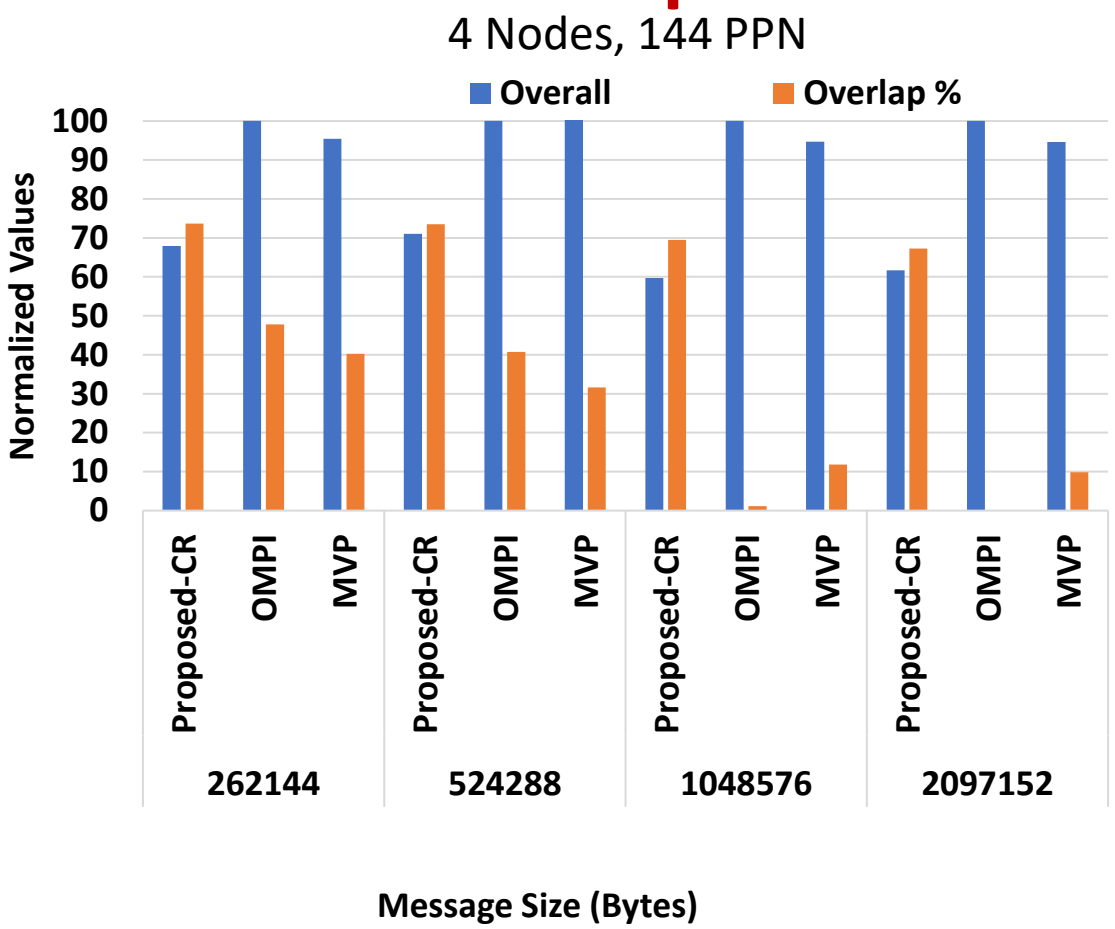
Results with miniAMR

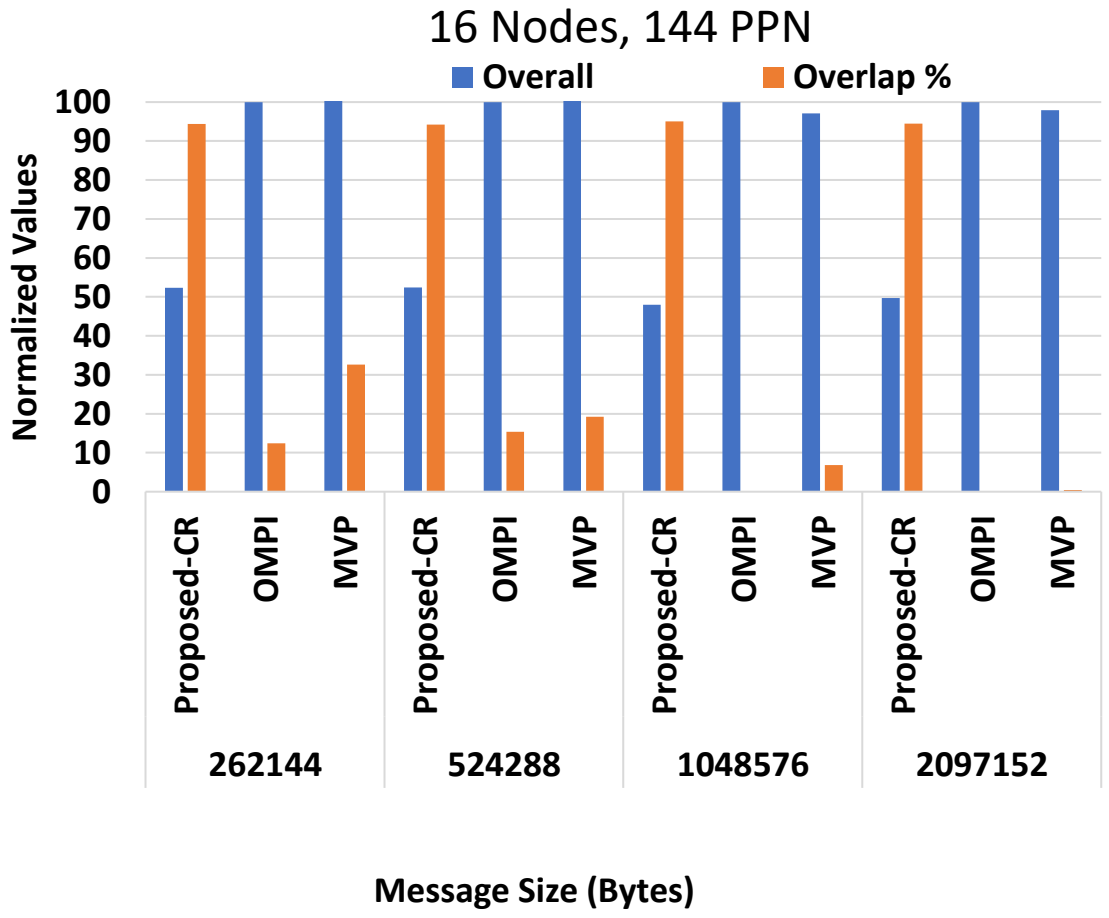




- Up to **37%** improvements for a fixed process count of 128 (and varying grid sizes)
- Up to **19.4%** for a fixed grid size of 32x32x32 (and varying process counts)

3D Stencil Overlap Benchmark





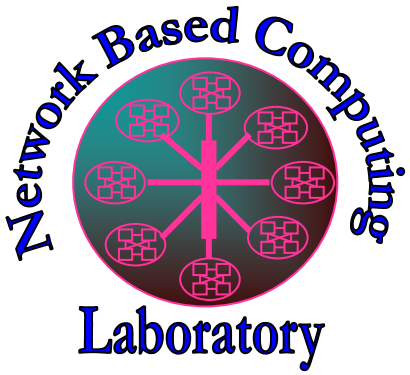
- Proposed scheme's overlap increases from **67% to 94%** as the scale increases for 2MB transfer
- Proposed scheme's overall latency performs from **32%** better for 256KB, **38%** better for 2MB on 4 Nodes
- For 2MB, the proposed scheme's overall latency is **38%** better on 4 Nodes, **50%** better on 16 Nodes

Table of Contents

Conclusion

- We identified significant shortcomings in existing MPI Rendezvous and work-stealing protocols for stencil-based communication patterns.
- We developed a dynamic intra-node protocol that adapts to application patterns by intelligently switching protocols and tuning chunk sizes at runtime
- These dynamic protocols lead to substantial performance gains and better overlap in HPC applications utilizing stencil communication, making them more efficient and scalable

THANK YOU!



Network-Based Computing Laboratory

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

The High-Performance MPI/PGAS Project

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

The High-Performance Big Data Project

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

The High-Performance Deep Learning Project

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