MVAPICH
MPI, PGAS and Hybrid MPI+PGAS Library

The Ohio State University

Efficient Designs for Distributed MPI Neighborhood Collectives

S. Mahdieh Ghazimirsaeed

Ghazimirsaeed.3@osu.edu

Postdoctoral Researcher
Department of Computer Science and Engineering
The Ohio State University

Outline

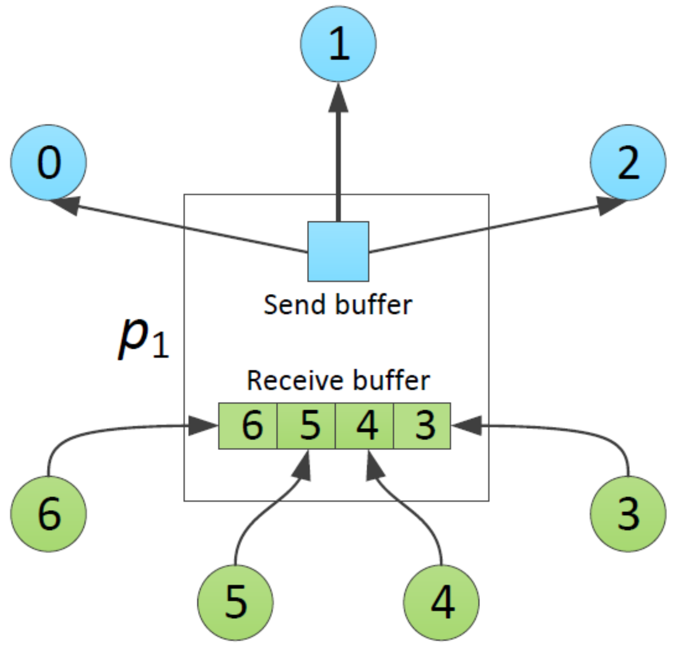
Introduction

- **HPC is the key in tackling the problems in different domains**
  - Communication is the major bottleneck in achieving performance

- **Message Passing Interface (MPI)**
  - Most popular parallel programming model in HPC
  - Provides communication APIs
    - Point-to-point
    - One-sided (RMA)
    - Collective (broadcast, all-to-all, etc.)
    - Neighborhood collective (allgather, alltoall, etc.)

Neighborhood Collectives

- Sparse collective communications
- Add communication functions to process topologies
  - Pattern is defined by the process topology graph
  - Only send/receive to/from neighbors
- Benefits:
  - User-defined collectives
  - Support for widely used patterns
  - Better scalability



Neighborhood Allgather

Can we design non-trivial algorithms to optimize neighborhood collectives?

Neighborhood Collectives – Prior Art

- **Well-known MPI libraries perform neighborhood collective operations using a naive approach**
- **Kandalla et al. designed neighborhood collectives using a network-offload approach**
- **Hoefler and Schneider proposed an algorithm that balances communications by offloading them from high-outdegree processes to those having lower outdegrees**
- **Traff et al. proposed a specification and message combining algorithms for isomorphic neighborhoods, with all processes having the same neighborhood structure.**
- **Mirsadeghi et al. proposed a pair-wise message combining mechanism for distributed graph topologies**

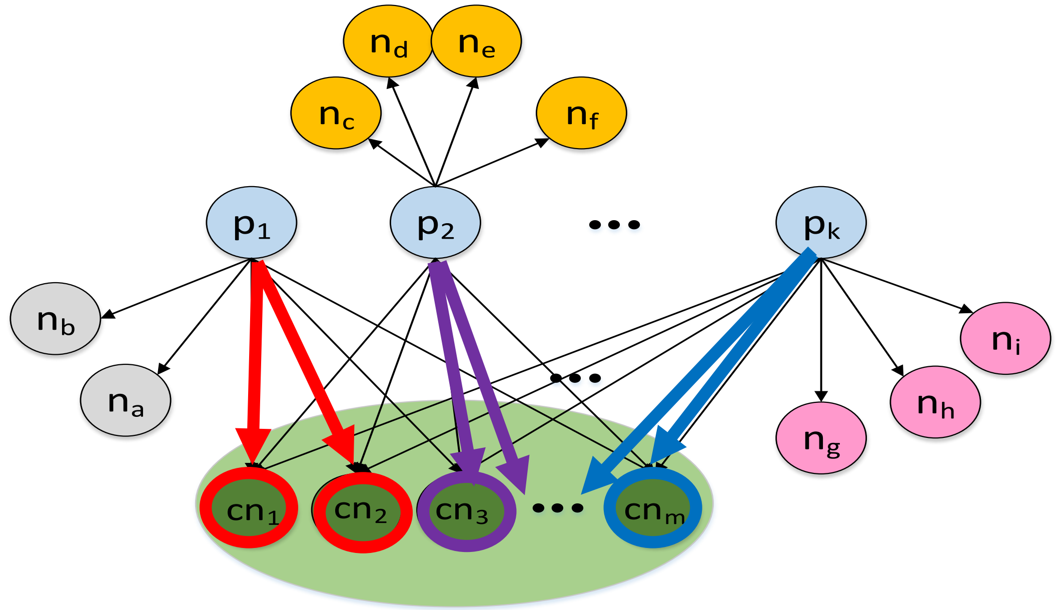
Prior work: Optimizing Neighborhood Collectives for Small Messages

- **Objective:**
  - Focus on small-message communications
    - Startup latency
      - Decrease the number of communication stages
  - Two design principles

1. Increase the number of senders in each stage
2. Increase the number of messages transferred in each send operation ➔ Message combining

Prior work: Optimizing Neighborhood Collectives for Small Messages

- **Exploit common neighborhoods for message combining**
- **Friend groups and common neighbors**
  - Classify the processes into friend groups of size k, which have more than θ neighbors in common
  - Each friend process is responsible for transferring combined messages to m/k of the common neighbors, after exchanging its message with the friends in the group



Optimizing Neighborhood Collectives for Large Messages

- **Fully distributed algorithm**

- **Using no prior knowledge about the topology**
  - Targeting MPI distributed graph topology

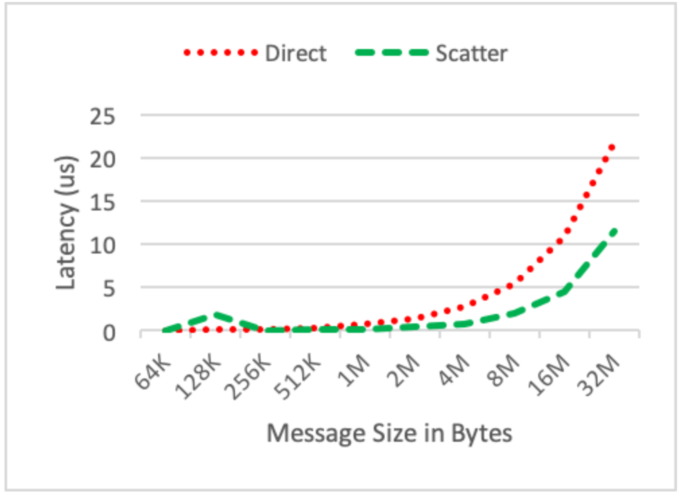
- **Focus on large messages**
  - Reduce the number of inter-node communication

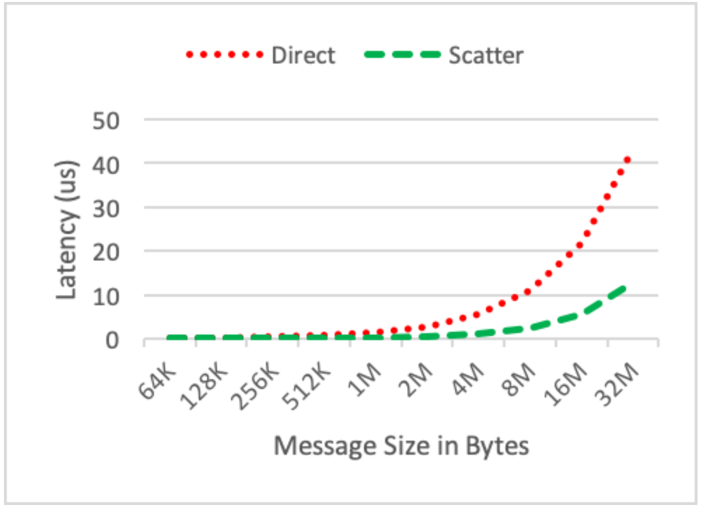
- **Two phases:**

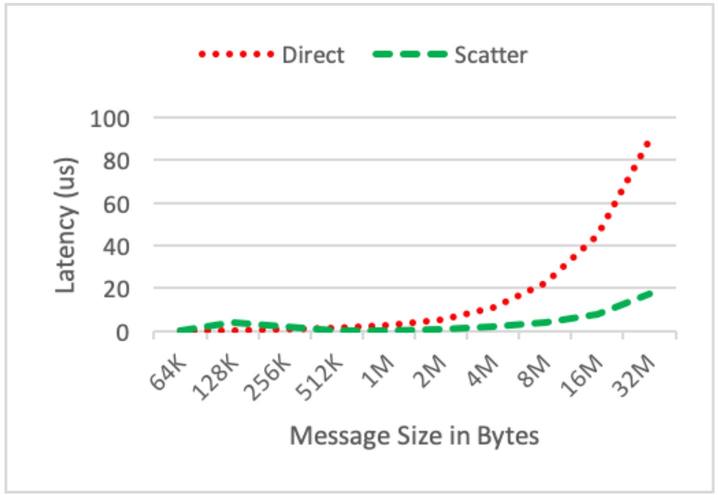
1. Communication Pattern design from the topology
2. Communication Schedule design from the communication pattern

Motivation

- **A process attempts to send the same message to multiple processes on the remote node:**
  1. sender process directly sends the data to the remote processes
  2. Sender chunks the large message and then it sends each chunk to remote processes. Then processes in the remote node perform an Allgather.







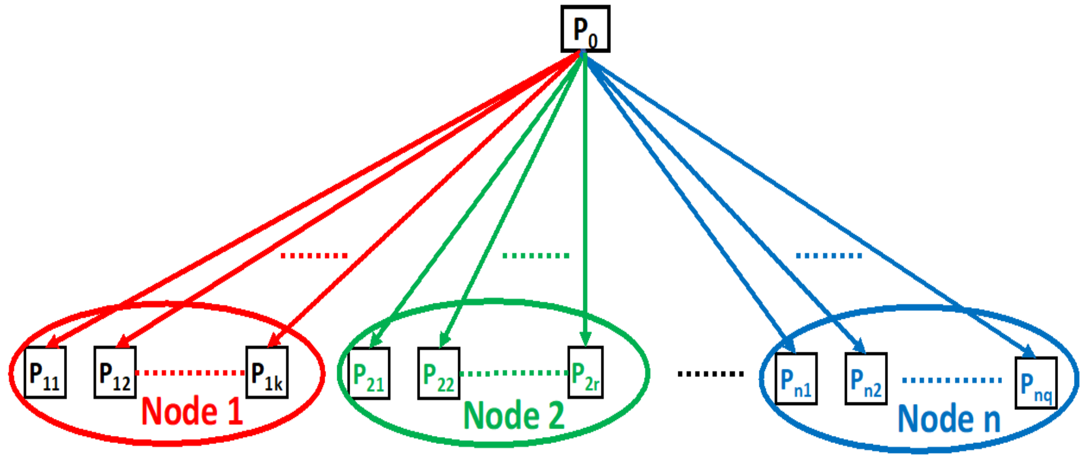
Motivation (Cont.)

- How can we redesign neighborhood collectives that take the underlying hardware architecture into account to achieve low latency?

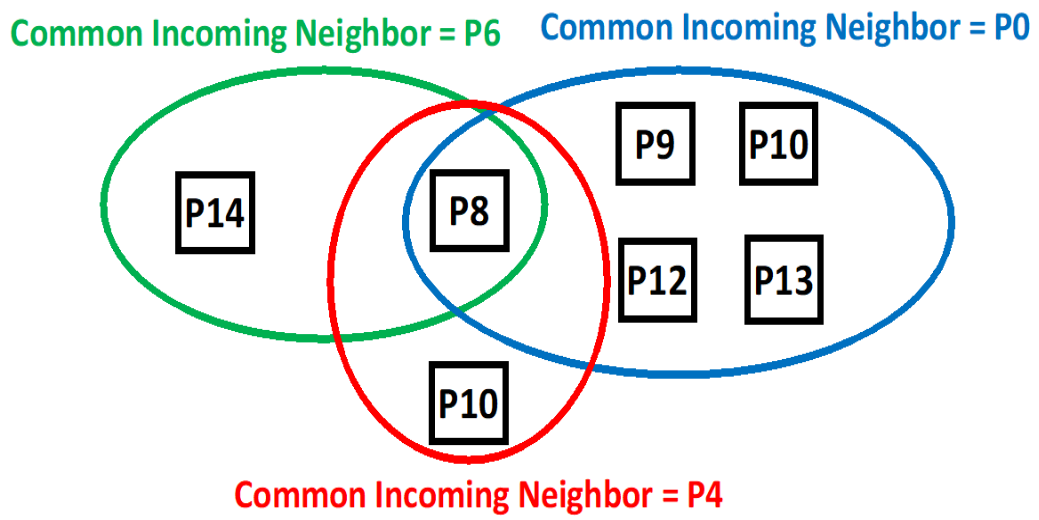
- How can a neighborhood collective efficiently distribute the data between remote processes, in other words, how can we design a neighborhood collective which improves load-balance between all the processes?

- How can we create communication patterns that locally describe the specific communications of each involved process and at the same time support any kind of process topologies?

The Proposed Communication Pattern Design



Part of a general virtual topology graph


Common Incoming Neighbor = P6
Common Incoming Neighbor = P0
P14
P8
P9
P10
P12
P13
P10
Common Incoming Neighbor = P4

The hypergraph associated with sample topology graph

The Proposed Communication Schedule Design

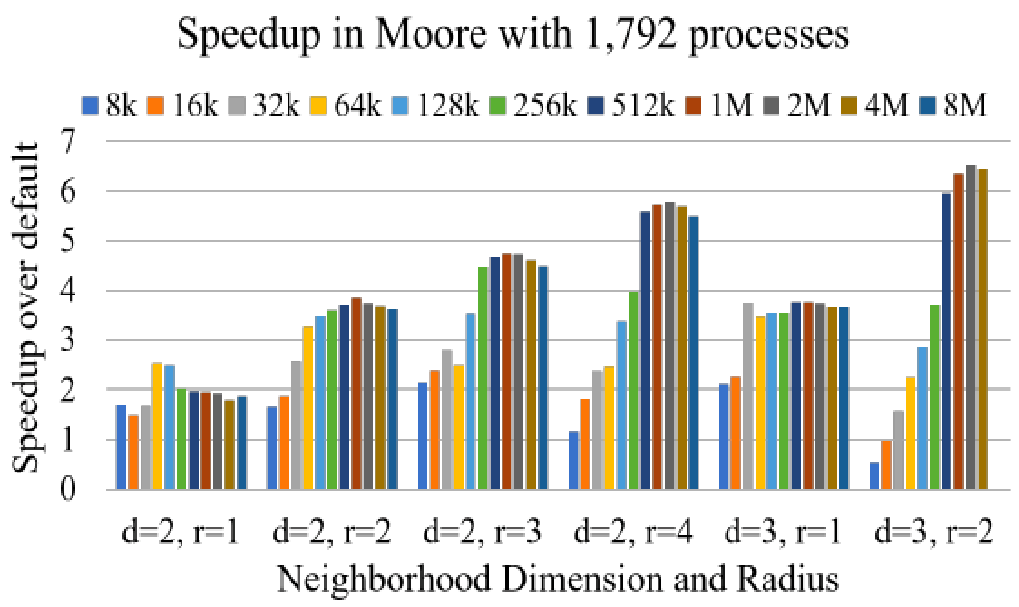
- Uses use the derived communication pattern to extract an efficient communication schedule for the neighborhood Allgather operation
- The communication pattern is created only once for each topology graph
- The communication schedule is used each time a neighborhood collective operation is called
- Three levels of communication:
  - *p* Communicates with intranode incoming and outgoing neighbors
  - *p* Scatter the data between Inter-node outgoing neighbors
  - Allgather between Inter-node outgoing neighbors of *p*

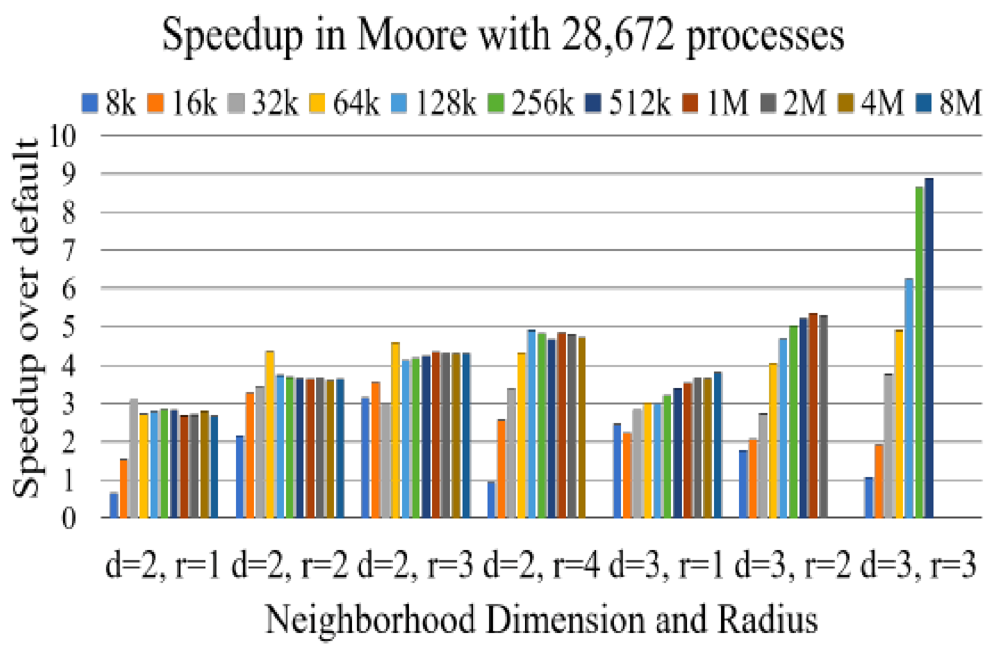
Experimental Setup

- **The InfiniBand-enabled Frontera cluster in Texas Advanced Computing Center**
  - 8008 compute nodes
  - Dual-socket Intel Xeon Platinum 8280
  - 56-core processors operating at 2.70 GHz with 192GB RAM
  - MVAPICH2-2.3.2

Microbenchmark Results and Analysis

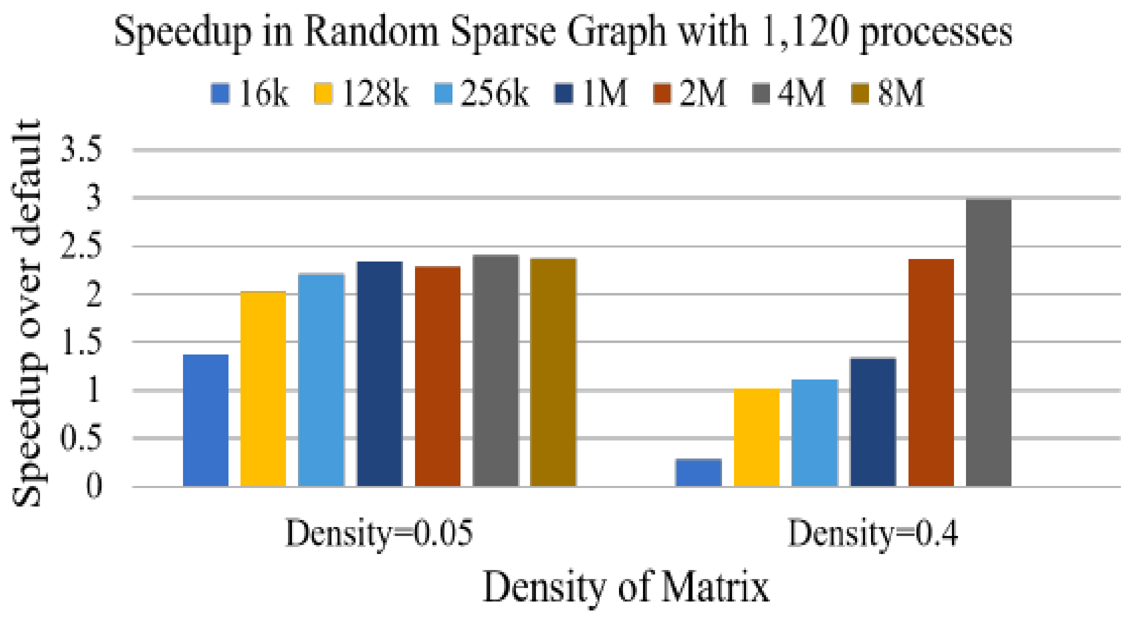
- **Moore Neighborhood**





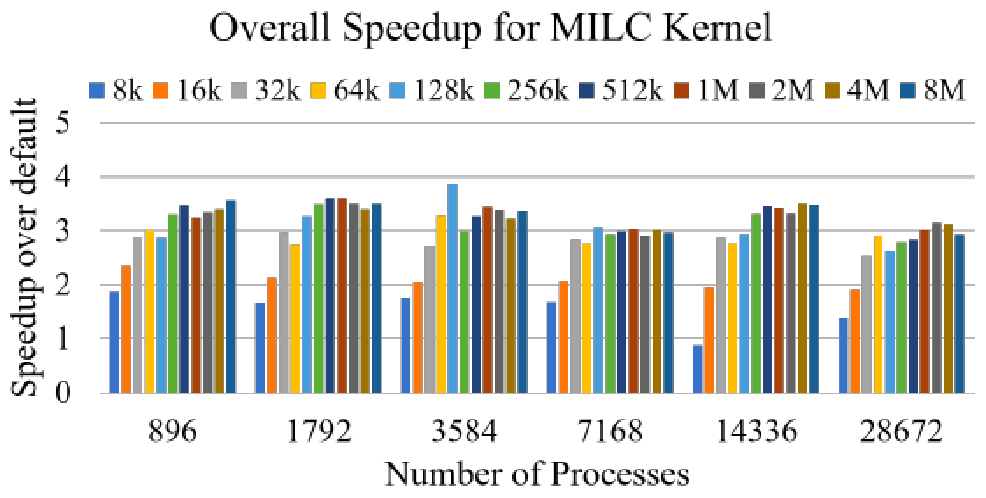
Microbenchmark Results and Analysis

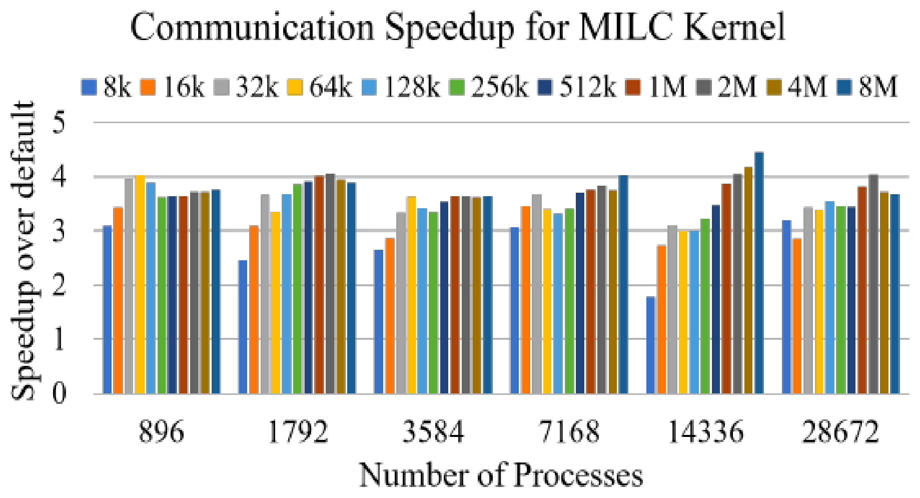
- **Random Sparse Graph**

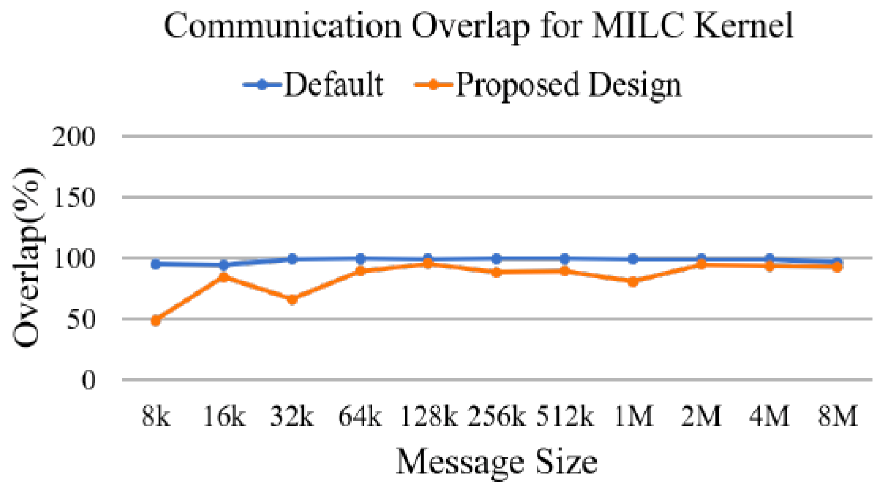


Application Results and Analysis

- **MILC Application Kernel**

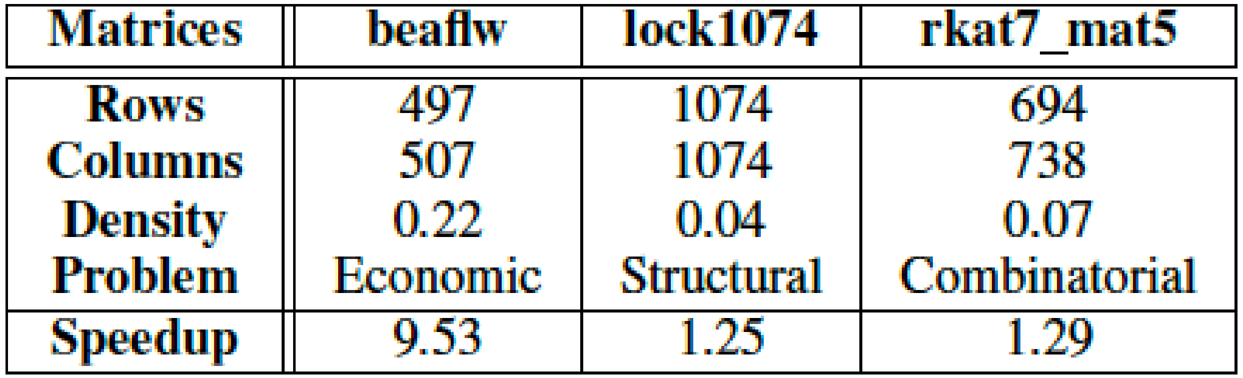






Application Results and Analysis

- **Sparse Matrix-Matirx Multiplication**



Matrices	beaflw	lock1074	rkat7_mat5
Rows	497	1074	694
Columns	507	1074	738
Density	0.22	0.04	0.07
Problem	Economic	Structural	Combinatorial
Speedup	9.53	1.25	1.29

Conclusion and Future Work

- **Conclusion**
  - Novel designs to efficiently increase the performance of large message neighborhood collectives
  - Modeling the communication pattern as a distributed hypergraph and determining the weight of hyperedges based on the virtual and physical topology of the processes
  - Load-aware: fair distribution of workload between processes
- **Future Work**
  - Add support for efficient communication for deep learning applications
  - Modify modify deep learning applications to take advantage of neighborhood collectives and consequently our design.