Design and Optimization of GPU-Aware MPI Allreduce Using Direct Sendrecv Communication

Presented at MUG '25

Chen-Chun Chen, Jinghan Yao, Hari Subramoni, and Dhabaleswar K. Panda

Department of Computer Science and Engineering

The Ohio State University

Outline

- Introduction
- Motivation
- Implementation and Optimization
- Evaluation Results
- Conclusion and Future Work

Outline

- **Introduction**
- Motivation
- Implementation and Optimization
- Evaluation Results
- Conclusion and Future Work

Introduction

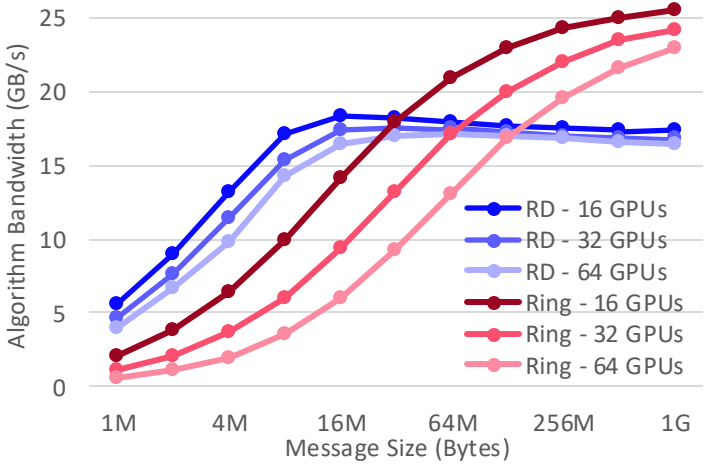
- Modern HPC and AI workloads rely heavily on **MPI_Allreduce** for aggregating data across thousands of GPUs.
- HPC applications:
  - Amber: a suite of biomolecular simulation programs, such as proteins, nucleic acids, and lipids.
    - Utilizes Allreduce communication for medium message sizes (MB)
- AI Applications/Frameworks:
  - Horovod with TensorFlow or PyTorch
  - PyTorch
    - Distributed Data Parallelism (DDP) framework for LLM/GPT-like models training
    - Utilizes both reduction-based and data-movement-based communications for medium to large message sizes (MB-GB)

Outline

- Introduction
- **Motivation**
- Implementation and Optimization
- Evaluation Results
- Conclusion and Future Work

Motivation

- Allreduce at scale remains challenging:
  Ring algorithm is good at large messages,
  but the **throughput drops as message is divided across more GPUs**.
- **Medium-sized messages** (MB range) fall into a performance gap.
- As the system scale increases, message sizes tend **to fall into the medium range**
  → inefficiencies



MPI Allreduce

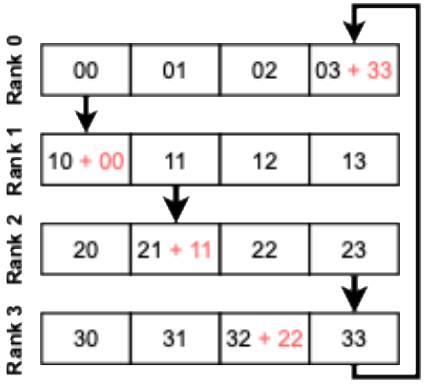
- Existing MPI collectives fail to utilize network bandwidth efficiently in this regime.
- How can we design a **GPU-aware Allreduce** that delivers low latency and **high bandwidth** for **medium messages** and **large-scale deployments** across **diverse** interconnects?

Outline

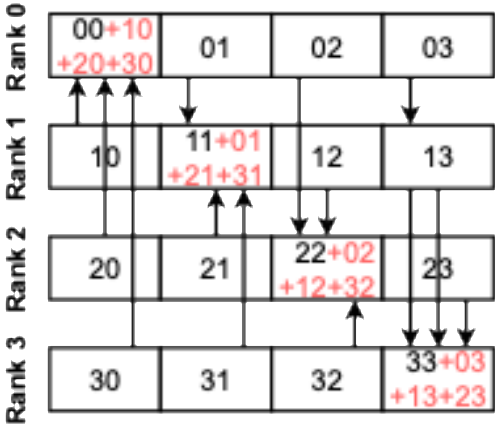
- Introduction
- Motivation
- **Implementation and Optimization**
- Evaluation Results
- Conclusion and Future Work

Proposed Direct Sendrecv Algorithm

- In modern MPI Allreduce implementations, the Reduce-Scatter-Allgather (RSA) algorithm is commonly adopted
  - Allreduce = Reduce-Scatter + Allgather
  - Reduce-Scatter: a distributed combine phase → Communication + Computation
  - Allgather: a collect phase → Communication
- Use **Direct Sendrecv algorithm** (inspired by the Alltoall communication pattern) to to redesign both the Reduce-Scatter and Allgather phases
  - Aims to **saturate** communication **bandwidth** and improve scalability by concurrently collecting messages from peer processes



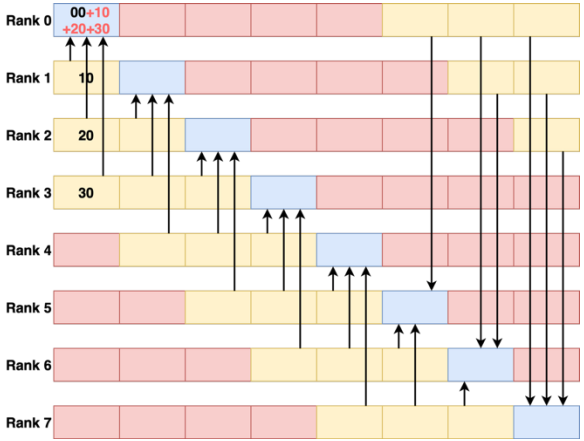
Ring Algorithm – The first step



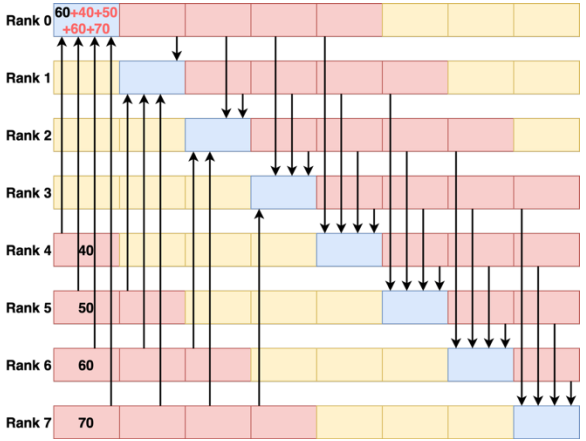
Direct Algorithm

Direct Reduce-Scatter with Throttling Designs

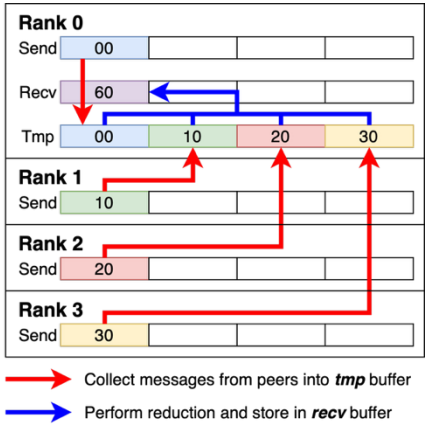
- Excessive concurrency may introduce overhead and eventually degrade performance.
- Throttling mechanisms:
  - Group **send** and **receive** calls into **smaller batches** to reduce contention.
  - Received data is staged in a *tmp* GPU buffer to enable the following computation step.
  - Example on processes using a throttle factor of 4 (2 communication batches):



Direct Algorithm with Throttling - Step 1



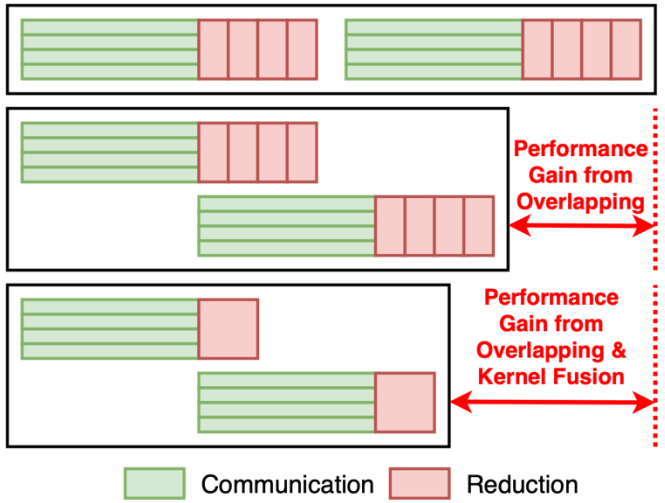
Direct Algorithm with Throttling - Step 2



Data Placement in RS Direct Algorithm

Optimization of Overlapping and Kernel Fusion

- In the Ring algorithms, element-wise communication and computation overlapping is not applicable.
- **No inter-iteration data dependencies** in the Direct algorithm:
  - Can be overlapped across batches.
  - **Overlapping**:
    - Change the blocking reduction calls **to non-blocking**.
    - Enqueue reduction calls on the same GPU stream.
    - A stream synchronization before Allgather phase.
- Launching multiple kernels still introduces overhead
  - E.g.: additional load/store operations to the *recv* buffer
  - **Kernel fusion strategy**:
    - **Combines** reduction tasks into a single kernel
    - Writes the final results directly to the *recv* buffer in **one** step



Timeline illustration comparing the proposed naive Direct design, Direct design with communication-computation overlapping, and the optimized version with both overlapping and kernel fusion.

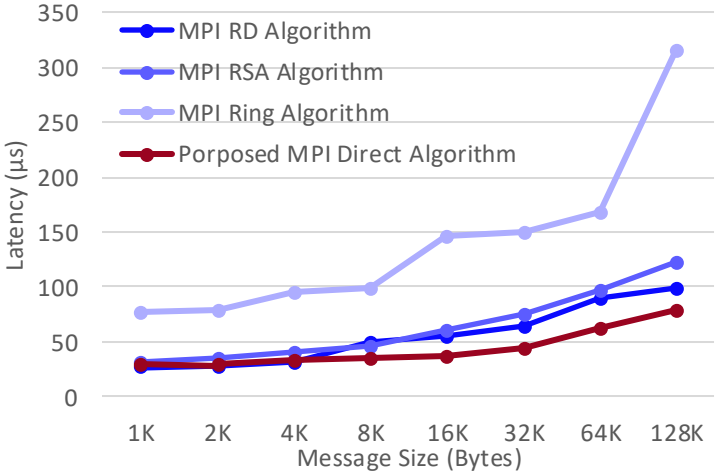
Outline

- Introduction
- Motivation
- Implementation and Optimization
- **Evaluation Results**
- Conclusion and Future Work

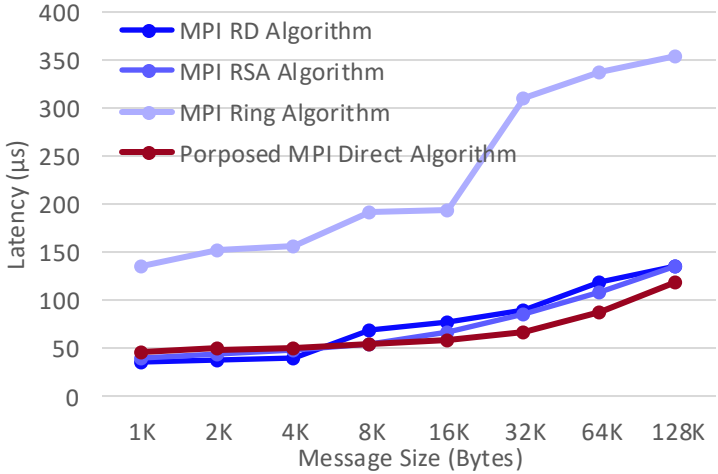
Evaluation Platform

Component	Vista	Cardinal	Delta-AI
GPU	1 NVIDIA GH200 GPUs	4 NVIDIA H100 GPUs	4 NVIDIA GH200 GPUs
Device Memory per GPU	96 GB HBM3e	96 GB HBM2e	96 GB HBM3e
CPU	NVIDIA Grace CPU	Intel Xeon Platinum 8470	NVIDIA Grace CPU
Memory	120 GB LPDDR5X	1 TB DDR5	480GB LPDDR5X
Sockets	1	2	4
Core per Sockets	72	52	72
Inter-connection	1 Infiniband NDR400 HCAs	4 Infiniband NDR400 HCAs	4 HPE Slingshot 200 Gbps NICs

Proposed Designs in CPU Staging for Small Messages



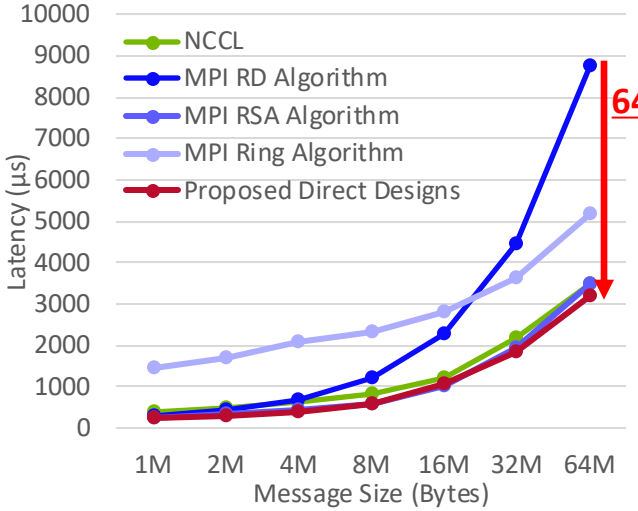
16 Vista Nodes (16 GPUs)



32 Vista Nodes (32 GPUs)

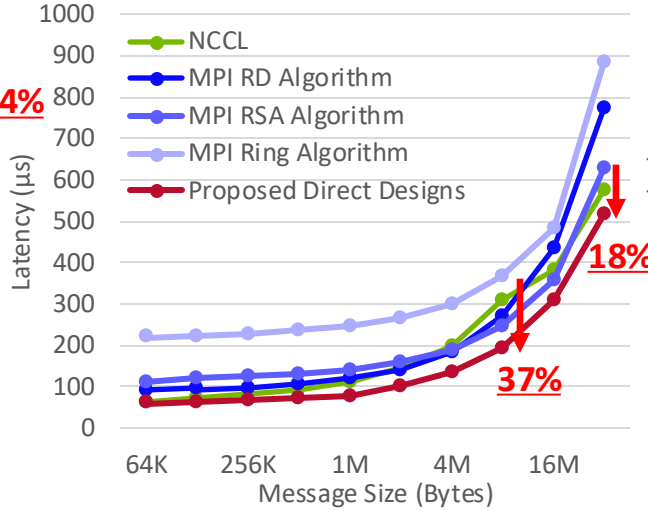
- We extend the proposed algorithm to the CPU side and integrate it into the CPU staging framework
- On 16 GPUs, our CPU-staged Direct algorithm outperforms the next-best method beyond **8 KB**
  - **34 µs** vs **46 µs (RSA)**, 49 µs (RS), 98 µs (Ring) at 8KB
- On 32 GPUs, it is better beyond **16 KB**
  - 57 µs vs 67 µs (RSA) at 16KB

Micro-Benchmark Evaluation



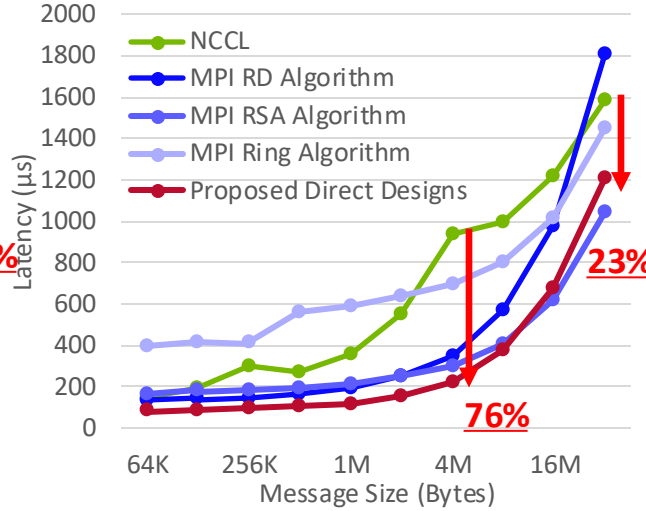
64 Vista Nodes (64 GPUs)

- **9%** lower latency than both NCCL and RSA, **38%** and **64%** improvements over Ring and RD at 64 MB
- **40%** lower latency than NCCL at 1 MB



8 Cardinal Nodes (32 GPUs)

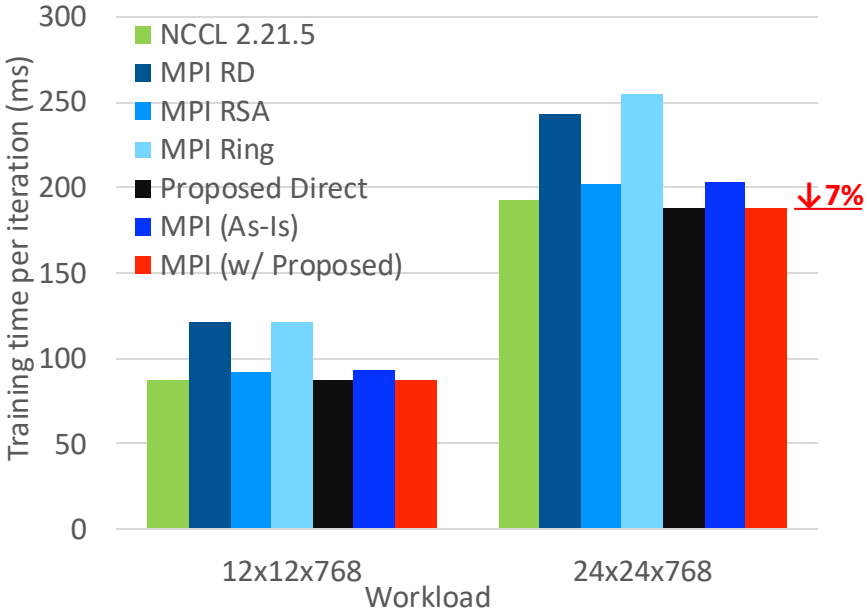
- **37%** improvement over NCCL at 8 MB
- Outperforms MPI RSA by **18%** and NCCL by **10%** at 32MB



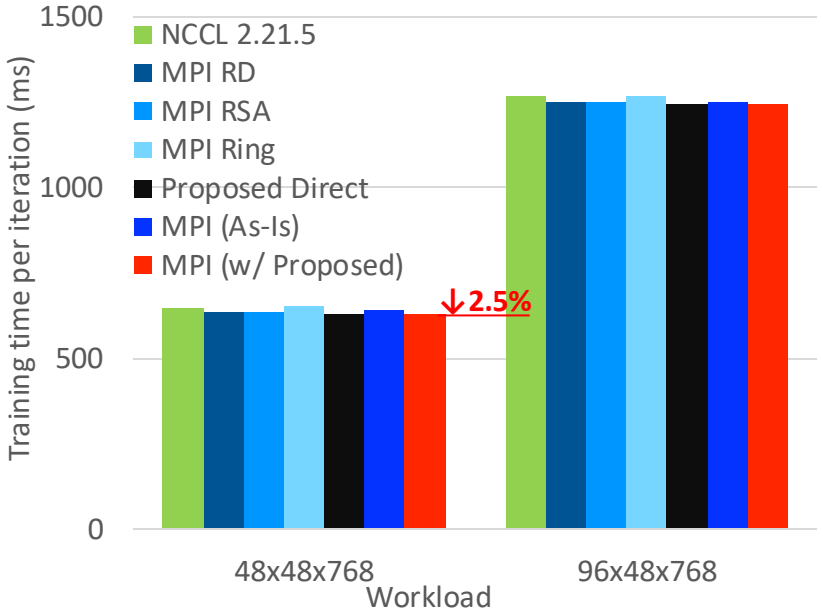
8 Delta-AI Nodes (32 GPUs)

- **30% to 40%** improvement over existing MPI algorithms and **30% to 76%** improvement over NCCL below 4MB
- **23%** reduction over NCCL at 32 MB

Application-level Evaluation: nanoGPT



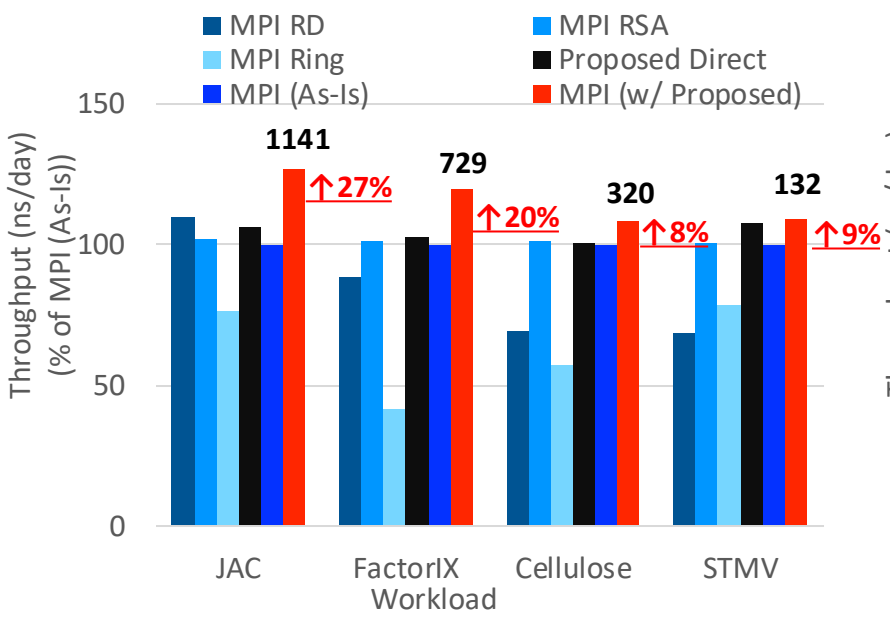
Vista – 64 nodes (64 GPUs)



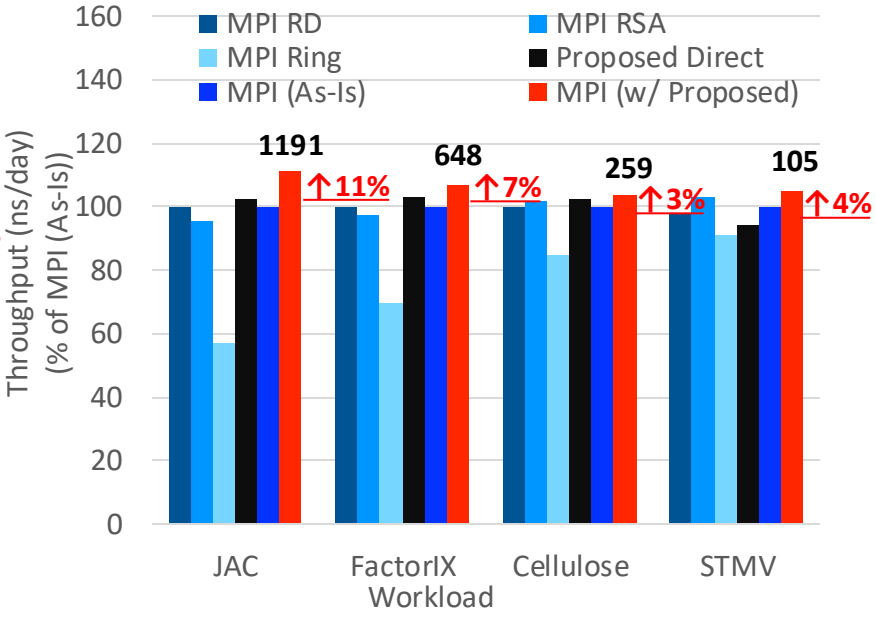
Cardinal – 8 nodes (32 GPUs)

- NanoGPT is a quick yet self-contained training and fine-tuning implementation for medium-sized GPTs.
- **7%** improvement over MPI(As-Is) on Vista and **2.5%** improvement over NCCL on Cardinal.

Application-level Evaluation: Amber



Vista – 32 nodes (32 GPUs)



Cardinal – 8 nodes (32 GPUs)

- Widely used biomolecular dynamics package, heavy use of Allreduce for forces, energies, etc.
- **27%**, **20%**, **8%**, and **9%** performance gain over MPI(As-Is) across the four workloads (on Vista).

Outline

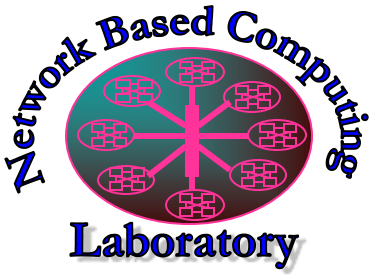
- Introduction
- Motivation
- Implementation and Optimization
- Evaluation Results
- **Conclusion and Future Work**

Conclusion and Future Work

- Allreduce operation at scale remains challenging, and existing MPI collectives fail to utilize network bandwidth efficiently in this regime.
- Proposed a **Direct Sendrecv Allreduce** with **throttling** to address the medium-message gap.
- Optimizations: **overlapping** communication & computation, **kernel fusion**, and **CPU staging**.
- Achieved up to **40%** lower latency vs NCCL and MPI baselines at medium sizes.
- Application-level benefits: **7%** faster **nanoGPT** training, **27%** faster **Amber** simulations.
- Demonstrated scalability across **diverse** GPU systems and interconnects.
- Available since MVAPICH-Plus 4.1rc (https://mvapich.cse.ohio-state.edu)
- Future plan: extend these designs to other collectives (Allgather, Reduce-Scatter, etc.)

Thank You!

panda@cse.ohio-state.edu



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/