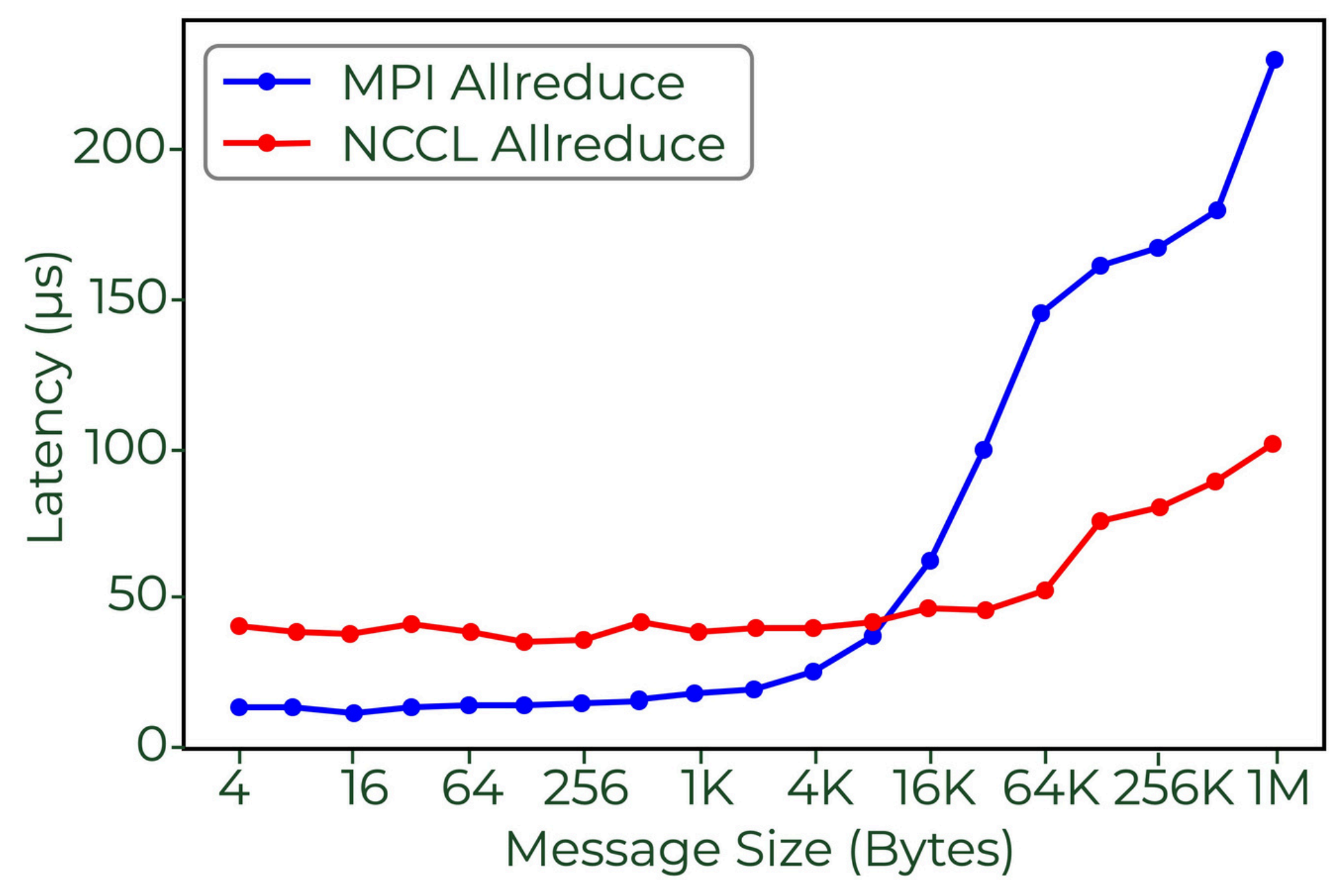
MPI4AI: RCCL Integration

PORTABLE COLLECTIVE COMMUNICATION FOR HETEROGENEOUS AI SYSTEMS

GRACE LI, DOE SULI Intern, Summer 2025
Mathematics and Computer Science Division, Argonne National Laboratory
Columbia University [Computer Science / Mathematics / Dance] | grace.li@anl.gov

Mentor: Dr. Mike Wilkins, Maria Goeppert Mayer Fellow
Mathematics and Computer Science Division
Argonne National Laboratory | wilkins@anl.gov



NCCL outperforms MPI for Allreduce at larger message sizes, motivating hybrid integration in this work. *Figure adapted from Chen, Su, Subramoni, & Panda (2023). https://doi.org/10.1145/3577193.3593724*

BACKGROUND & MOTIVATION

- MPICH is Argonne's implementation of the MPI (Message Passing Interface) standard, and scalable collective communication is essential for high-performace computing (HPC) and artificial intelligence (AI) workloads
- Collective operations coordinate parallel data exchange and aggregation across multiple processes
- Graphics processing unit (GPU) collective communications libraries (CCLs) like NVIDIA CCL (NCCL) and ROCm CCL (RCCL) offer performance optimizations, but at the cost of portability and standard compliance
- A 2023 OSU study showed that hybrid MPI-xCCL approaches outperform either backend alone
- Therefore, we aim to integrate RCCL, AMD's ROCm-based collective library, into MPICH, building on the existing NCCL Allreduce backend to enable hybrid execution for optimal performance

My LinkedIn!

PROCESS

1. Implemented RCCL-specific Allreduce logic
2. Integrated RCCL into MPICH's algorithm selection framework
3. Enabled runtime RCCL selection via environment variables
4. Added auto-detection of RCCL and HIP in MPICH build configuration
5. Benchmarked collective composition strategies (alpha, beta, gamma)
6. Evaluated hybrid MPI-RCCL performance with custom logic file

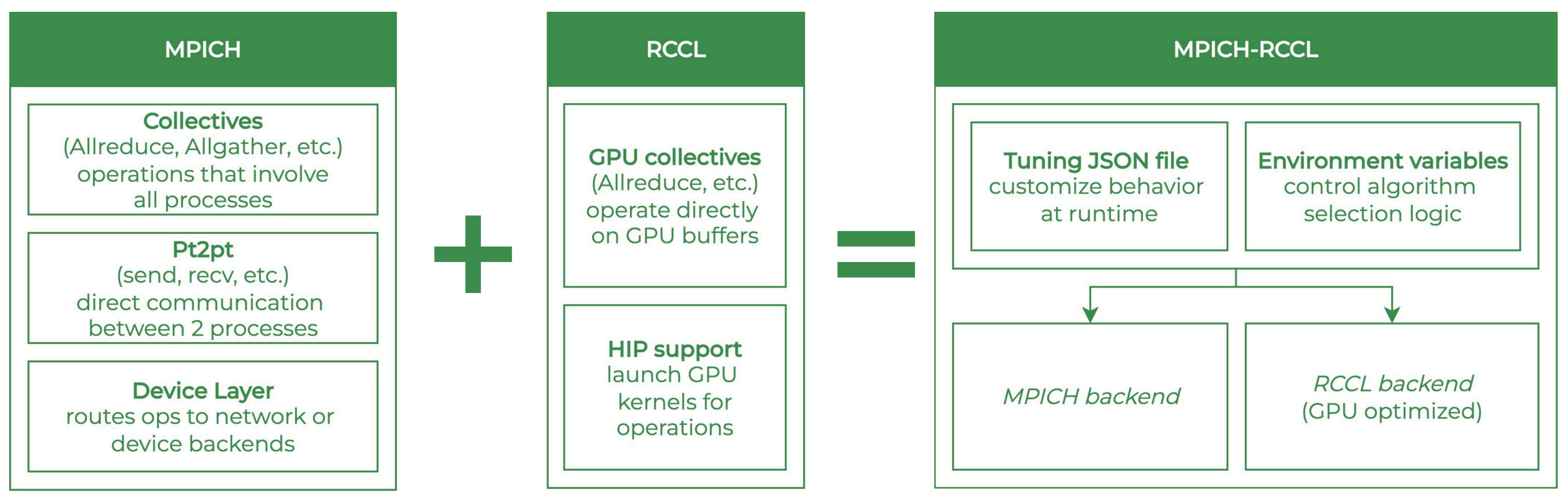
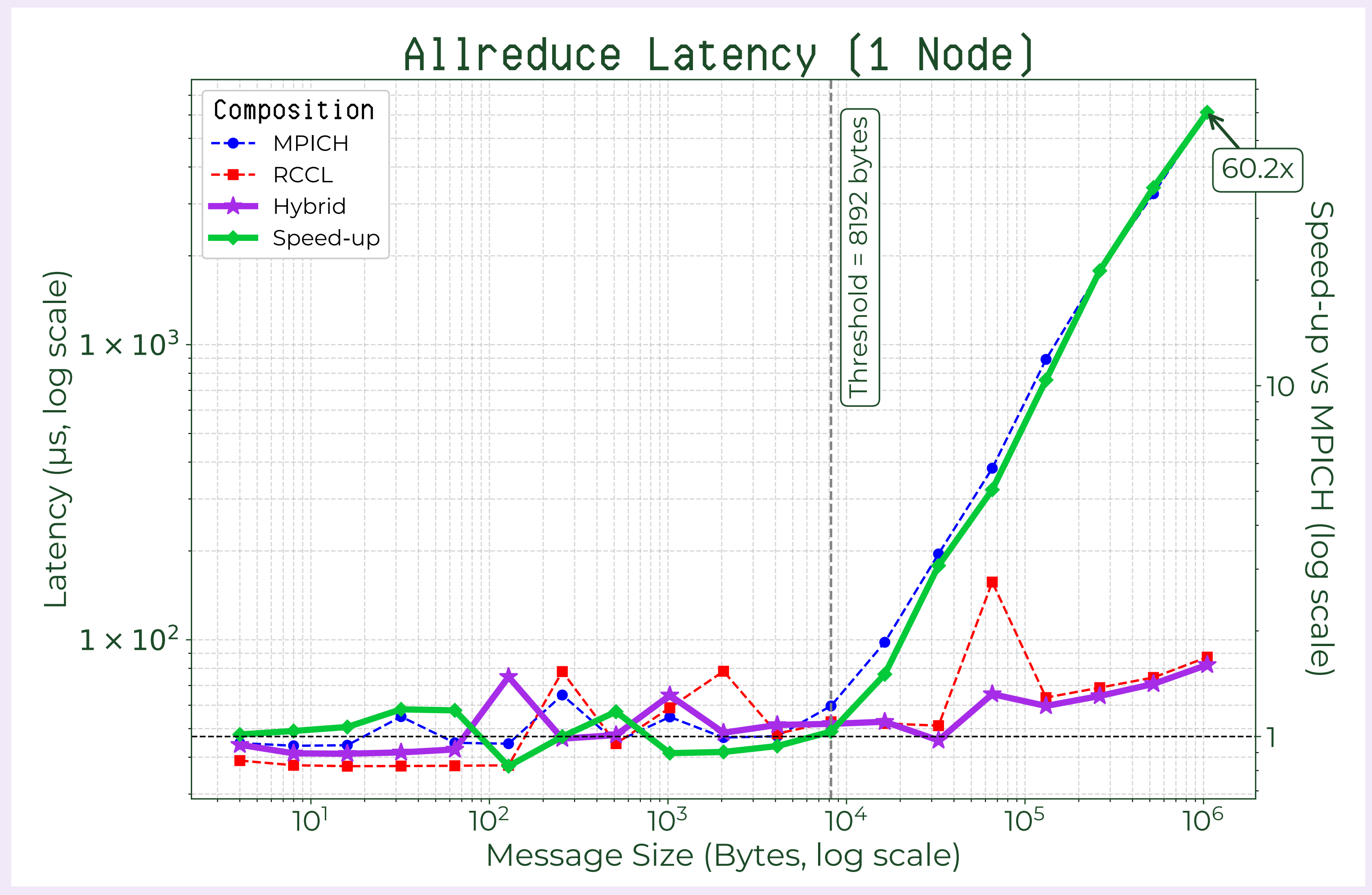

MPICH
Collectives
(Allreduce, Allgather, etc.) operations that involve all processes
Pt2pt
(send, recv, etc.) direct communication between 2 processes
Device Layer
routes ops to network or device backends
+
RCCL
GPU collectives
(Allreduce, etc.) operate directly on GPU buffers
HIP support
launch GPU kernels for operations
=
MPICH-RCCL
Tuning JSON file
customize behavior at runtime
Environment variables
control algorithm selection logic
MPICH backend
RCCL backend
(GPU optimized)

Figure: Allreduce latency for different backend compositions on 1-node, 2-node, and Frontier setups. Lower latency indicates better performance. The purple line switches from MPI to RCCL at the threshold, yielding significant gains at larger message sizes. The green line shows speedup over the baseline (higher is better).



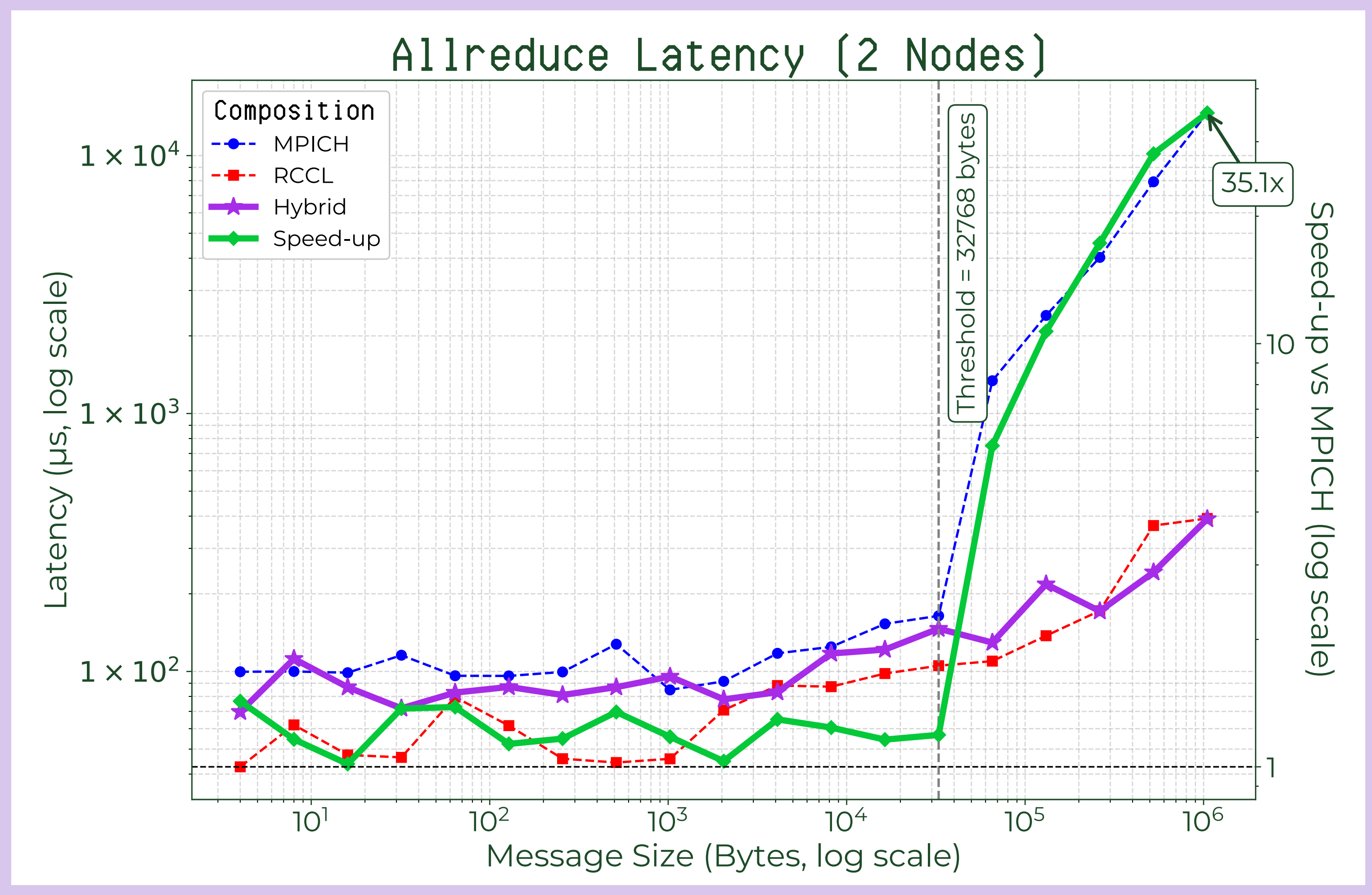
EVALUATION

methodology

- OSU Micro-Benchmarks measure latency across a range of message sizes (4 – 1048576 bytes)
- Evaluated on JLSE nodes (4 AMD GPUs per node)
- Tested MPICH & RCCL individually with device collectives disabled or enabled
  - Using alpha, beta, & gamma algorithms
- Hybrid logic controlled by custom tuning file & environment variables

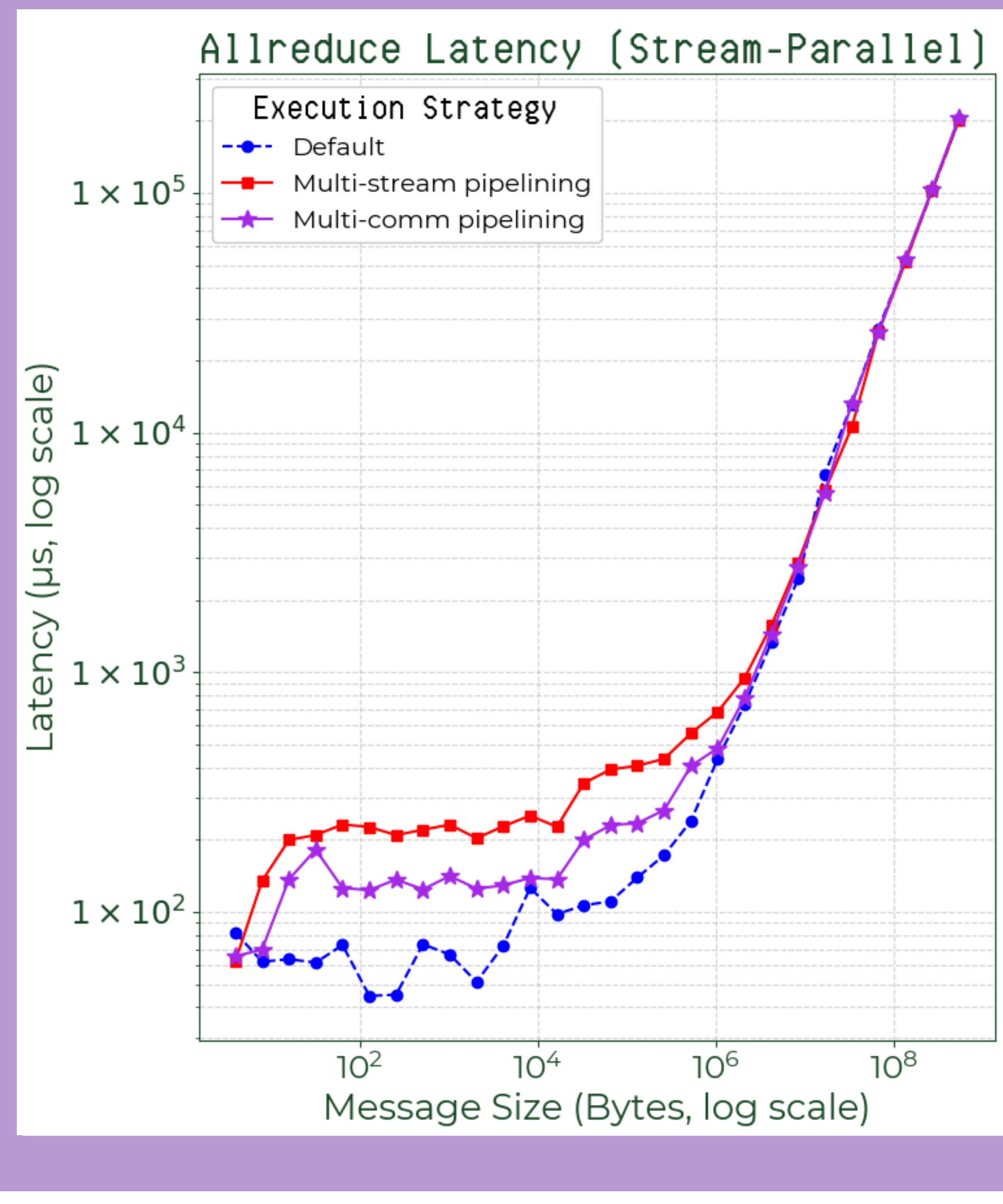
results

- RCCL significantly outperforms MPICH, especially for larger message sizes
- Single Node (4 GPUs): 8KB threshold
  - Switches from mpi-gamma to rccl-beta
  - RCCL is 60× faster than MPICH at 1MB
- Two Nodes (8 GPUs): 32KB threshold
  - Switches from mpi-alpha to rccl-beta
  - RCCL is 35× faster than MPICH at 1MB



CONCLUSION

- Integrating AMD's RCCL backend into MPICH delivers significant performance gains for collective operations, especially for large message sizes where GPU acceleration is most effective
- The hybrid MPICH-RCCL backend retains MPI's portability and standard compliance
- Dynamic backend switching enables intelligent selection between CPU and GPU collectives
- This work advances the MPI4AI vision by demonstrating how MPI can evolve to remain performant and relevant for AI workloads in modern, GPU-accelerated computing environments



Pipelining

- Single-stream Allreduce underutilizes GPU concurrency for large messages
- Modern GPUs support multiple streams and asynchronous execution
- Goal: overlap communication using multiple streams to reduce latency
- Builds a foundation for future hybrid and pipelined collective strategies
- **1 communicator, 4 streams**: message is chunked into 4 parts, each reduced on a separate GPU stream
- **4 communicators, 4 streams**: each chunk is assigned to a separate RCCL communicator, each with its own stream
- This is a work in progress, and we will continue refining the design and implementation to better utilize stream-level concurrency and reduce latency

FUTURE WORK

- Plan to run multi-node experiments on Frontier at Oak Ridge to understand scaling behavior
- Extend support to other collective operations (e.g., Broadcast, Allgather, Scatter)
- Contribute PR upstream for community adoption

acknowledgements

This project was made possible thanks to the mentorship of Dr. Mike Wilkins and the support of my collaborators, Nihar and Sean. I'm grateful to the MPICH/PMRS team for their foundational work, including the initial NCCL backend integration, which laid the groundwork for this effort. This work was supported in part by the U.S. Department of Energy, Office of Science, Office of Workforce Development for Teachers and Scientists (WDTS) under the Science Undergraduate Laboratory Internships (SULI) program. Special thanks to Argonne National Laboratory for this research opportunity and Oak Ridge National Laboratory for providing access to Frontier to run scaling experiments.

Citation: Chen, A., Su, K., Subramoni, H., & Panda, D. K. (2023). *MPI-xCCL: A portable MPI library over collective communication libraries for various accelerators. In Proceedings of the ACM International Conference on Supercomputing (ICS '23)* (pp. 205–216). Association for Computing Machinery. https://doi.org/10.1145/3577193.3593724

MPI4AI: RCCL Integration

by Grace Li | grace.li@anl.gov

Mentor: Dr. Mike Wilkins | wilkins@anl.gov

Message Passing Interface (MPI) remains the core standard for distributed communication in high-performance computing, and is widely used across scientific domains. However, as AI and scientific workloads increasingly rely on GPUs, traditional CPU-based collectives in MPI become bottlenecks. Libraries like NVIDIA's NCCL and AMD's RCCL offer better performance through direct GPU memory access, but lack portability, limiting integration into existing applications. Therefore, this project integrates RCCL into MPICH, Argonne's reference MPI implementation, enabling tunable, dynamic, message-size-aware switching between CPU and GPU backends. The initial implementation supports the Allreduce operation, with plans to extend support to additional collectives. Small-scale testing demonstrates over 60× lower latency for large messages (>1 MB) on a single node and over 35× on two nodes compared to native MPICH. This work advances the MPI4AI initiative by enhancing MPI's relevance for heterogeneous, accelerator-driven environments and enabling efficient communication for AI training and other GPU workloads.