Design and Implementation of a GPU-Aware MPI Collective Library for Intel GPUs

Chen-Chun Chen, Goutham Kalikrishna Reddy Kuncham, Hari Subramoni and Dhabaleswar K. (DK) Panda
The Ohio State University
{chen.10252, kuncham.2}@osu.edu, {subramon, panda}@cse.ohio-state.edu

Research Motivation

- Accelerators/GPUs and the inter-connections are pivotal in contemporary HPC ecosystems.
- Despite entering the GPU market later, Intel is actively involved in the design and development of various GPU products and their associated ecosystems.
- GPU-aware MPI libraries have performed well with NVIDIA and AMD systems, and the same is expected for Intel GPUs.

Research Challenges

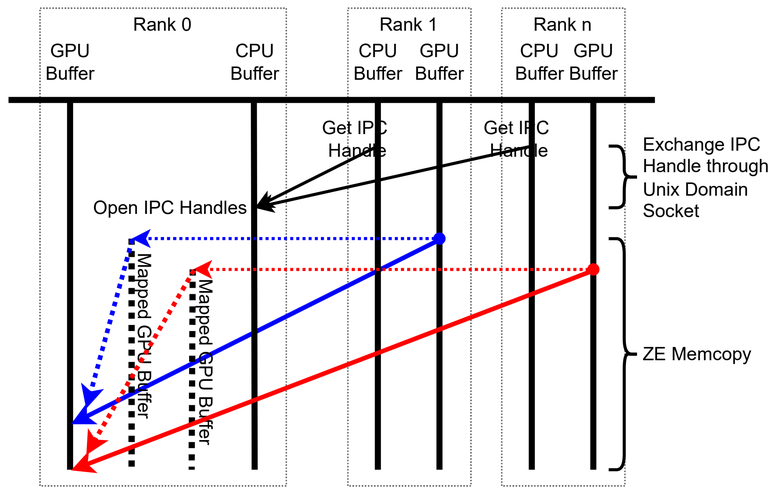
- How can we design a library for intra-node communication across multiple GPUs, leveraging the high speed and bandwidth of Intel X^e links?
- How can we design a library for inter-node communication across multiple nodes?
- How can we design a collective library for both data-movement- and reduction-based operations?

Overview of the Designs

- **Data-movement operations**: previous work primarily utilized simple point-to-point send/recv operations for collective communication, which introduced additional overhead.
- **Reduction operations**: previous work simply utilized CPU staging approaches to offload the device data to CPU buffers and performed CPU-based reduction.
- In this work:
  - We adopt a hybrid approach to deal with different message size ranges.
  - **Data-movement operations**: we focus on large message communication and utilize IPC techniques.
  - **Reduction operations**: we focus on large message communication and computation by utilizing IPC and kernel-based techniques.
  - For the inter-nide Allreduce operation, we implement a two-level algorithm to fully exploit the benefits of our kernel designs.

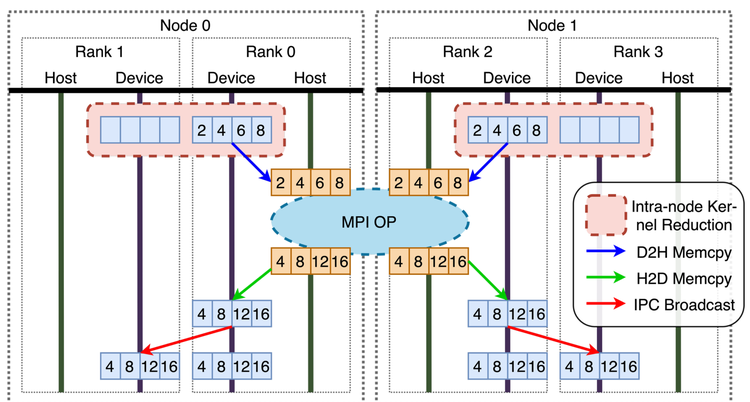
IPC Designs for Large Messages

- As-is: rely on launching multiple point-to-point calls.
- To-do: aggregate multiple memory copy commands into a single operation.
- Avoid redundant launches and synchronizations.
- Consider the entire data pattern as a holistic picture.

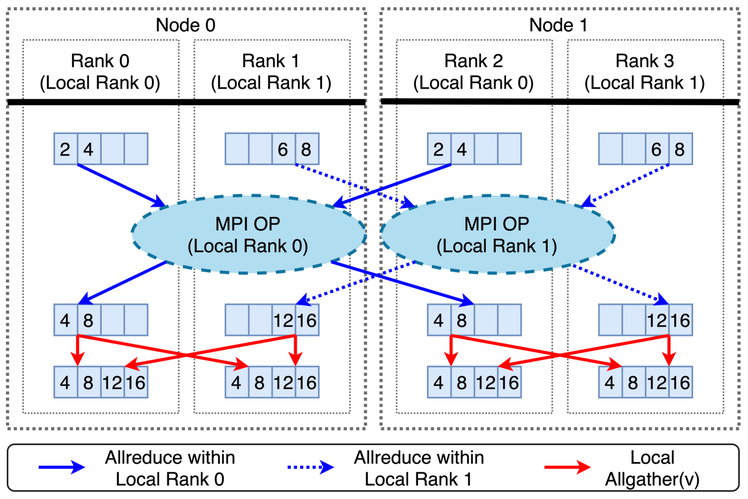


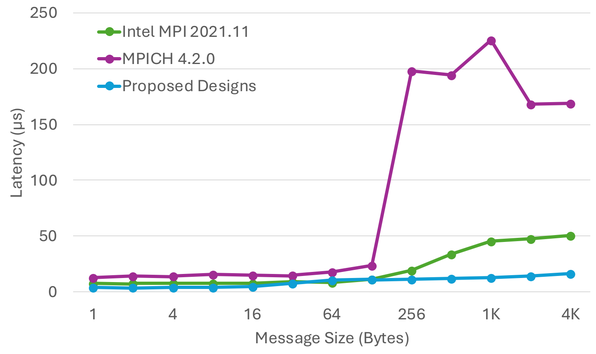
Inter-node Allreduce Implementations

- **Single-leader 2-level Allreduce**:
- 1st-level: intra-node reduction, performed by the kernel.
  - Store the data on **local rank 0**, the local leader rank, rather than on each GPU.
- 2nd-level: inter-node reduction
  - With the data residing in the leader ranks of each node, only one process per node participates in this operation.
  - CPU-based leader Allreduce with CPU staging techniques.



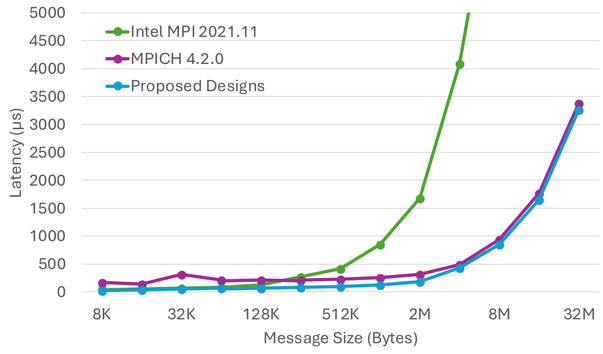
- **Multi-leader 2-level Allreduce**:
  - Each local rank group handles an equal portion of the data to distribute the workload evenly.
- The reduction kernel store the temporary results on **each GPU**.
- Each group performs a similar inter-node leader Allreduce among the involved processes
- Perform a local Allgather(v) communication to dispatch the final reduction data.





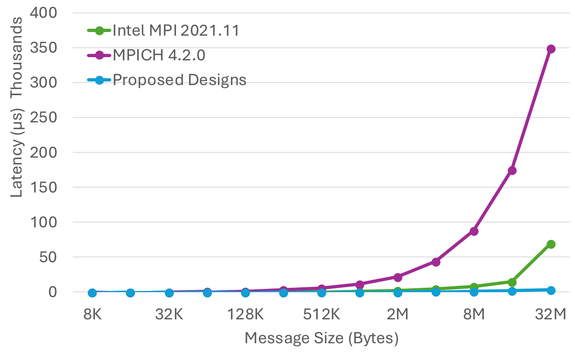
Alltoall - Small Messages - 1 Node (4 GPUs)

- Lowest latency **4 µs**, compared to 8 µs using Intel MPI and 15 µs using MPICH.



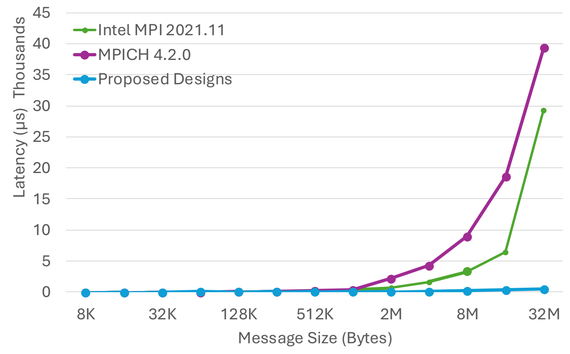
Alltoall - Large Messages - 1 Node (4 GPUs)

- Aachieve a latency of 3258 µs at 32 MB, **21x** faster compared to Intel MPI.



Alltoallv - Large Messages - 1 Node (4 GPUs)

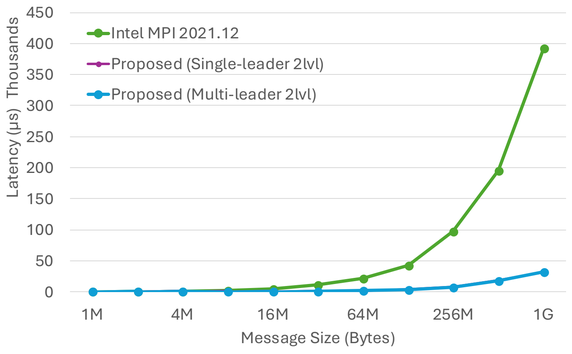
- Similar trend to our Alltoall performance, but **100x** faster compared to Intel MPI.



Bcast - Large Messgaes - 1 Node (4 GPUs)

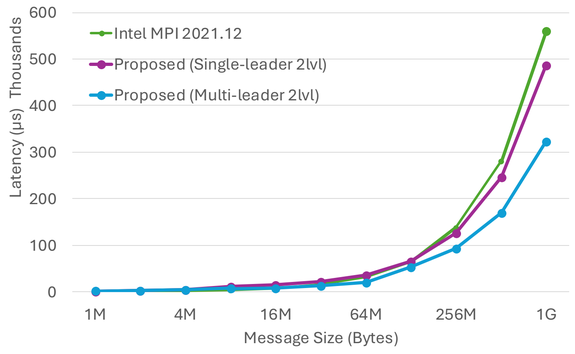
- **53x** faster and **71x** faster compared to Intel MPI and MPICH.

Benchmark-level Performance Evaluations - Reduction Collectives



Allreduce - 1 Node (8 GPUs)

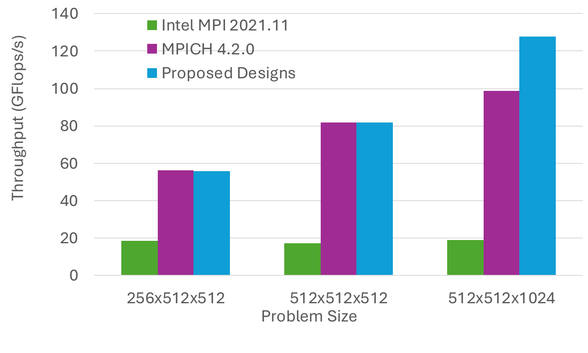
- On 1 node, **92%** improvement (11x faster) at 1 GB message size.



Allreduce - 4 Node (32 GPUs)

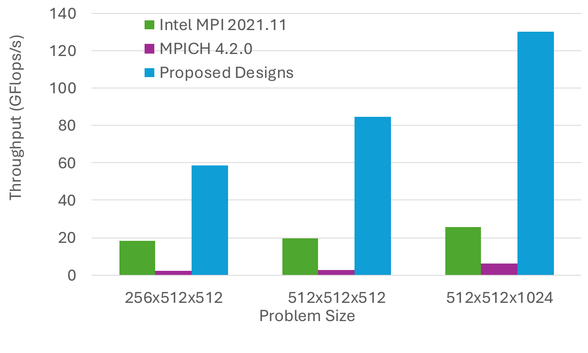
- **13%** improvement with the proposed **single-leader 2-level** design.
- **42%** improvement with the proposed **multi-leader 2-level** design.

Application-level Performance Evaluations - heFFTe



heFFTe with Alltoall - 1 Node (4 GPUs)

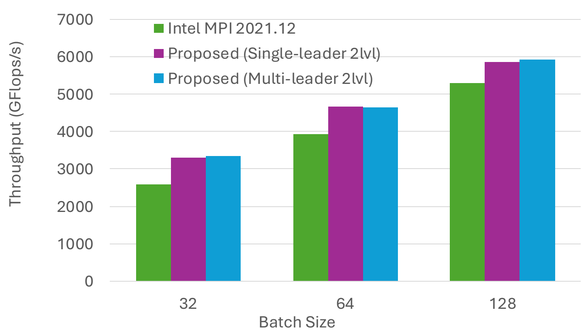
- **128 GFlops/s** in the 512x512x1024 case, surpassing MPICH's 98 GFlops/s by **1.3x**.



heFFTe with Alltoallv - 1 Node (4 GPUs)

- **20~33** times compared to MPICH.
- **5** times compared to Intel MPI.

Application-level Performance Evaluations - Deep Learning Applications



TensorFlow + Horovod - 4 Node (16 GPUs)

- **22%** of performance improvement at batch size of 32

PyTorch + Horovod - 4 Node (16 GPUs)

- **28%** of performance improvement at batch size of 32

Conclusion

- Utilized IPC-based techniques to enhance data-movement collective performance, and combined kernel-based and IPC-based approaches to optimize both intra- and inter-node performance for reduction collectives.
- **Benchmark results**:
  - **100x** improvement in Alltoallv operations for large messages compared to MPICH.
  - **42%** improvement in Allreduce (1 GB, 16 GPUs) over Intel MPI.
- **Application-level impact**:
  - Up to **33x** improvement for heFFTe
  - **28%** for PyTorch with Horovod.
- **Available in MVAPICH-Plus 4.0** (https://mvapich.cse.ohio-state.edu)

MVAPICH

THE OHIO STATE UNIVERSITY

References
1. C. Chen, G. Kuncham, P. Kousha, H. Subramoni, D. Panda: "Design and Implementation of an IPC-based Collective MPI Library for Intel GPUs", PEARC24
2. C. Chen, G. Kuncham, H. Subramoni, D. Panda: "Design and Implementation of Kernel-based MPI Reduction Operations for Intel GPUs", HiPC24

Acknowledgements
- NSF grants #1818253, #1854828, #2007991, #2018627, #2311830, #2312927, and XRAC grant #NCR-130002