EMMA: Efficient Multi-node Memory-aware AllReduce Algorithms

Valentino Guerrini, Ke Fan, Sidharth Kumar

vguer24@uic.edu kefan23@uic.edu sidharth@uic.edu

ABSTRACT

AllReduce is a **critical collective** in both HPC and large-scale AI workloads. However, scaling it to Exascale systems presents key challenges due to **inter-node communication bottlenecks** and **underutilization** of intra-node resources like **shared memory** and **NVLink**.

This work analyzes state-of-the-art AllReduce algorithms to identify **inefficiencies** and **opportunities** for hybrid strategies that explicitly **separate intra- and inter-node** communication. We introduce a preliminary algorithmic design that leverages **tunable** intra-node communication patterns and discuss key performance criteria, including **message count** and **data volume**.

Our early results provide insight into **communication trade-offs** and guide the development of **adaptive AllReduce implementations** optimized for Exascale systems.

1 - CONTEXT & MOTIVATION

AllReduce is a collective communication operation that combines data from all processes and distributes the result back to all. It's widely used in:

- **HPC workloads** (e.g., distributed linear algebra, simulations)
- **AI training** (e.g., synchronizing gradients across GPUs)

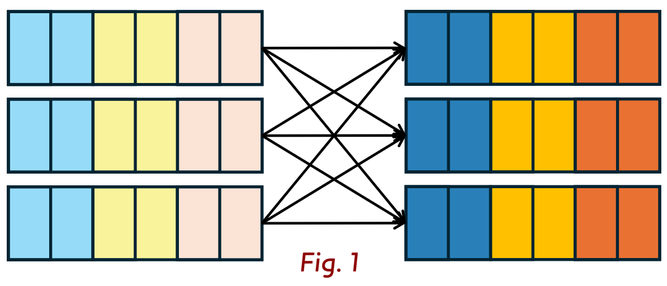


Fig. 1

Formally, each process contributes a buffer of values, and a reduction operator (e.g., sum, max) is applied element-wise across all buffers. The result is then broadcast back to every process.

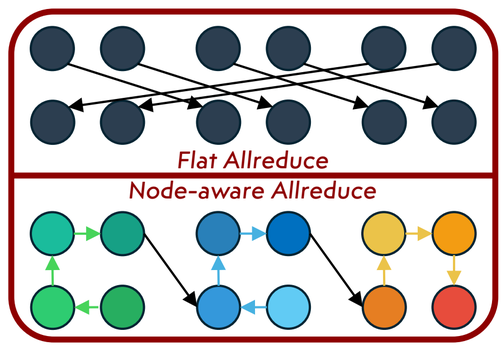
Why it matters: AllReduce is latency-sensitive and bandwidth-bound, making it a major performance bottleneck in large-scale systems. As compute nodes become faster and more parallel, communication overhead — especially **inter-node communication** — increasingly dominates.

Challenge at Exascale:
Exascale systems feature:

- **Tens of thousands of nodes**
- **Deep memory hierarchies**
- **Fast intra-node links** (e.g., shared memory, NVLink)
- **Slower inter-node links** (e.g., Ethernet, InfiniBand)

However, most *Current AllReduce algorithms treat all communication **uniformly**, often failing to take full advantage of **high-bandwidth intra-node resources***.

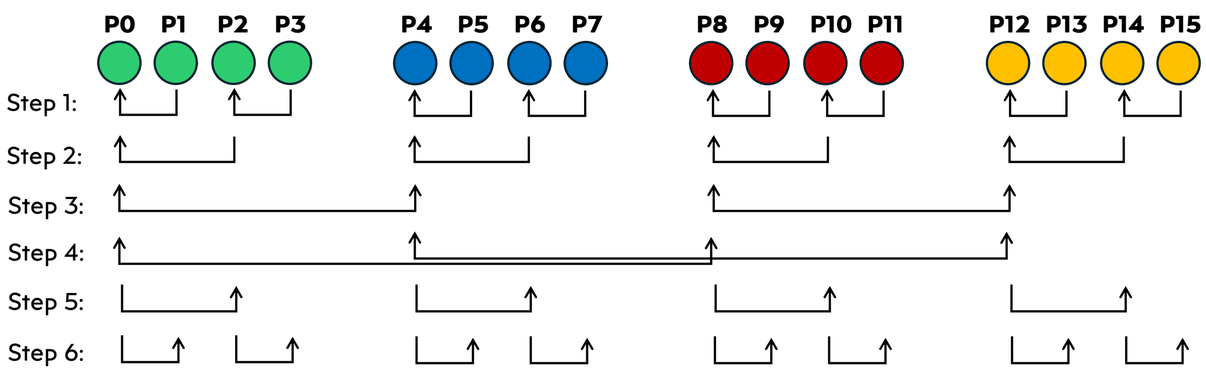
Fig. 2



3 – TOPOLOGY AWARE STRATEGIES

While most default AllReduce implementations treat communication uniformly, several works have begun to explore **topology-aware** designs that explicitly separate intra- and inter-node phases.

SMP-Aware Collectives (e.g., MVAPICH, OpenMPI, MPICH with SHM)

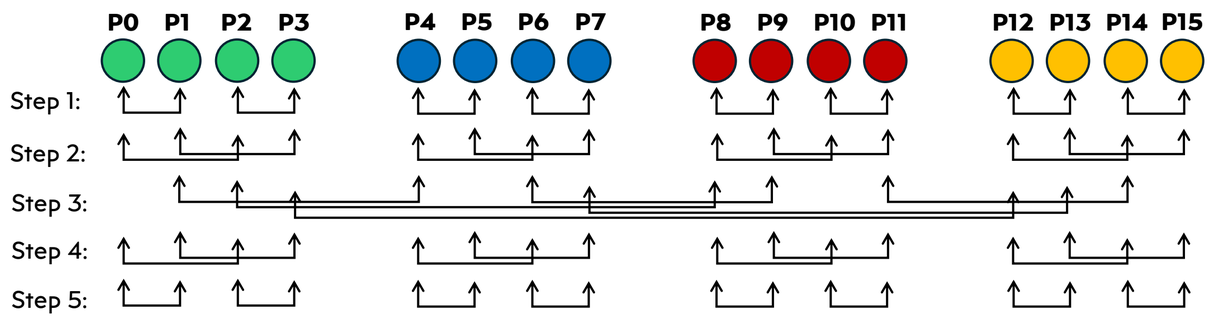


These implementations typically:

- Use **shared memory** for intra-node reductions
- Perform a **leader-based inter-node exchange**
- Then **broadcast** the result within each node

This *node-aware reduction pipeline* improves efficiency by leveraging faster local communication, but often remains tightly coupled to **specific system architectures** and is **not tunable**.

Amanda Bienz et al. [2]



Improves intra-node utilization through a **multi-leader strategy**, where **multiple ranks per node** participate in the inter-node phase. However still lacks **runtime tunability** and assumes relatively **uniform node configuration**.

CONCLUSIONS

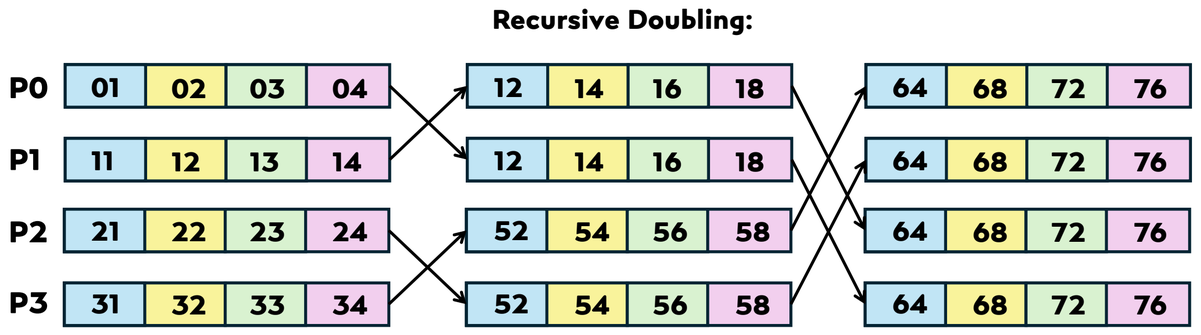
We presented EMMA, a **tunable** and **topology-aware** AllReduce algorithm that generalizes existing hierarchical approaches by introducing **flexible parameters**. Our initial single-node results demonstrate **performance gains** over both MPICH and Amanda Bienz's strategy **for small messages**. Ongoing work focuses on **scaling** EMMA across nodes, refining its hierarchical structure, and enabling runtime adaptability.

2 – PROBLEM ANALYSIS

While AllReduce is widely used, its performance depends heavily on message size, process count, and system topology. Most MPI libraries use a small set of algorithms selected through fixed, non-adaptive heuristics.

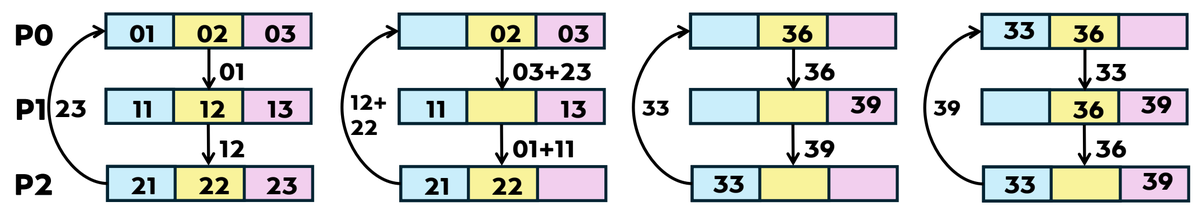
Common Algorithms:

Recursive Doubling:



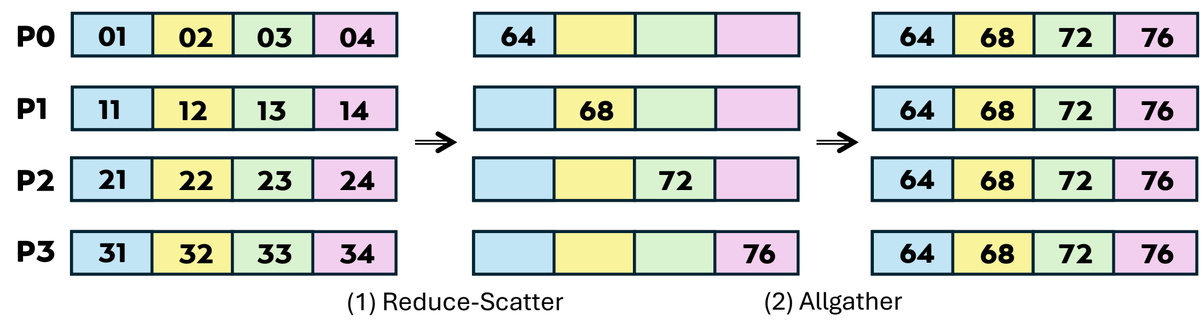
Efficient for small messages. Scales as log(P), but doesn't differentiate between intra- and inter-node steps.

Ring:



Good bandwidth usage but slow for small messages and high latency.

Rabenseifner[1]:



Combines Reduce-Scatter and Allgather phases. More efficient for medium-sized messages, but assumes homogeneous communication cost.

Limitations :

- Algorithms are often **hard-coded** with thresholds and **non-adaptive**.
- In practice, **MPICH/OpenMPI switch between algorithms** at fixed message sizes without considering node topology or memory hierarchy.
- Many do **not exploit intra-node shared memory** for local reduction or reuse.
- There's a lack of **parametrizable strategies** that can adapt to system and workload characteristics.

4 – EARLY DESIGN AND RESULTS

We introduce **EMMA** (*Efficient Memory-aware Multi-level AllReduce*), a hierarchical algorithm designed to optimize collective communication on modern multi-node systems. EMMA explicitly separates intra- and inter-node communication and is **parametrized by**:

- r: **Radix**, which defines the base of the communication tree (i.e., number of peers per round) in both Reduce-Scatter and Allgather phases
- b: **Block count**, which controls how many logical data blocks are processed per round — typically tuned to match the number of local processes or shared-memory characteristics

EMMA generalizes Rabenseifner's **two-phase** structure by allowing **full control** over communication **granularity** and **structure**. This enables adaptation to system topology, message size, and bandwidth asymmetry.

Progress so far:

- Implemented **Reduce-Scatter** inspired by Tuna2[3] with tunable r and b
- Implemented **Allgather** using r-Bruck
- Verified correctness across test cases and process counts

We conducted our initial evaluation of EMMA on a single node of Polaris, a leadership-class supercomputer at Argonne National Laboratory

Peak Performance: 34 petaflops

Architecture: AMD EPYC "Milan" processor

Cores: 17,920

Network Switch: 200 Gbps

In this configuration, EMMA **matches MPICH** for small messages (≤ 512 doubles) and is consistently **Faster than Amanda Bienz's** hierarchical algorithm in early multi-node experiments

Next Steps:

- Scale EMMA to **512 processes** (8 nodes) and evaluate parameter effects
- Extend Allgather with **batching support** for 2-layer hierarchical r-Bruck
- Explore alternative designs for both algorithm phases

REFERENCES:
[1] Rajeev Thakur, Rolf Rabenseifner, and William Gropp. 2005. Optimization of collective communication operations in MPICH. The International Journal of High Performance Computing Applications 19, 1 (2005), 49–66.
[2] A. Bienz, L. N. Olson, and W. D. Gropp, "Node-aware improvements to allreduce," 2019. [Online].
[3] K. Fan, J. Domke, S. Ba, and S. Kumar, "Configurable non-uniform all-to-all algorithms," 2024. [Online].