ACCELERATING COLLECTIVE COMMUNICATION IN DATA PARALLEL TRAINING ACROSS DEEP LEARNING FRAMEWORKS

JOSH ROMERO, DEVELOPER TECHNOLOGY ENGINEER

NSDI 22

MOTIVATION

HPC and DL

Over the past few years, the usage of DL within an HPC/scientific computing context has exploded in popularity:

- Science codes generate a lot of simulation data and DL can enable advanced data analytics

HPC centers historically are home to the largest supercomputing clusters, purpose-built for performant scaling of science programs from hundreds to thousands of CPUs/GPUs (e.g., 27,000 V100 GPUs on the Summit supercomputer):

- To take advantage of these extreme scale resources for DL, we require DL framework communication libraries that can scale as well as any well-tuned science code on these types of systems.

HOROVOD

Framework for Distributed Deep Learning

Horovod is a framework for distributed deep learning:

- Originally developed by Uber but now a project within the Linux Foundation AI.
- **Simple:** just a few lines of modifications to existing single worker scripts
- **Framework-agnostic:** works with TensorFlow, PyTorch, and MXNet

LF AI

HOROVOD

Framework for Distributed Deep Learning

Key challenge: multiple worker processes of frameworks using graph-based operation scheduling will not submit gradient tensors for reduction in consistent orders

→ To avoid deadlocking, a mechanism to coordinate scheduling of collective operations needs to be established.

LF AI

HOROVOD

Core Design and Operation

So how does Horovod coordinate the collectives?

- Core design relies on a **control/coordination layer (“control plane”)** that processes incoming requests (“metadata”) from all workers and determines what collectives to perform and when.
- Uses a “coordinator-worker” design, where one worker tasked with most decision-making logic. Control messages transferred across the network with MPI or Gloo.
- **Operates dynamically**: coordination logic runs in a background thread that checks for new tensors in a fixed tic rate (called a cycle)

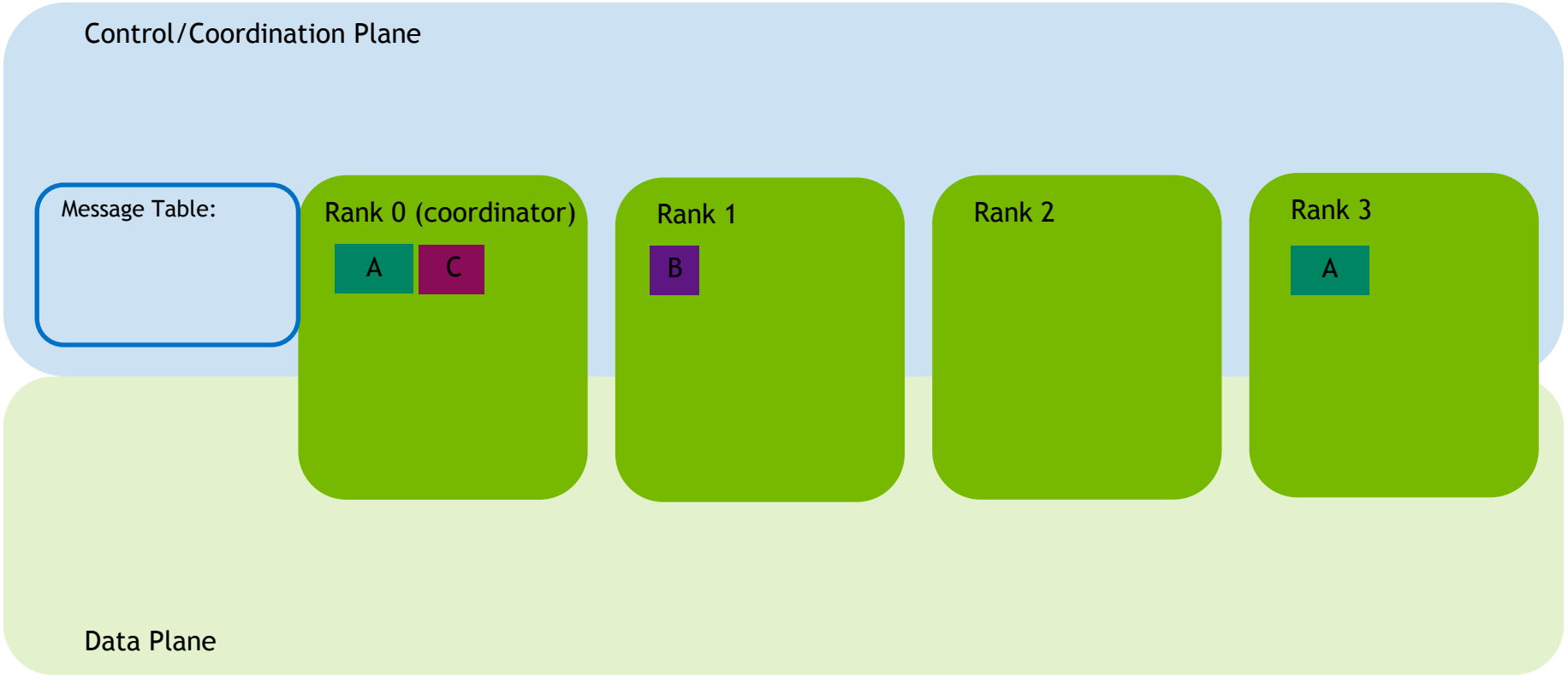
The actual collective operations (allreduce, allgather, broadcast) carried out on the “**data plane**” with MPI, NCCL, or Gloo.

COORDINATION PROCESS

Original Design

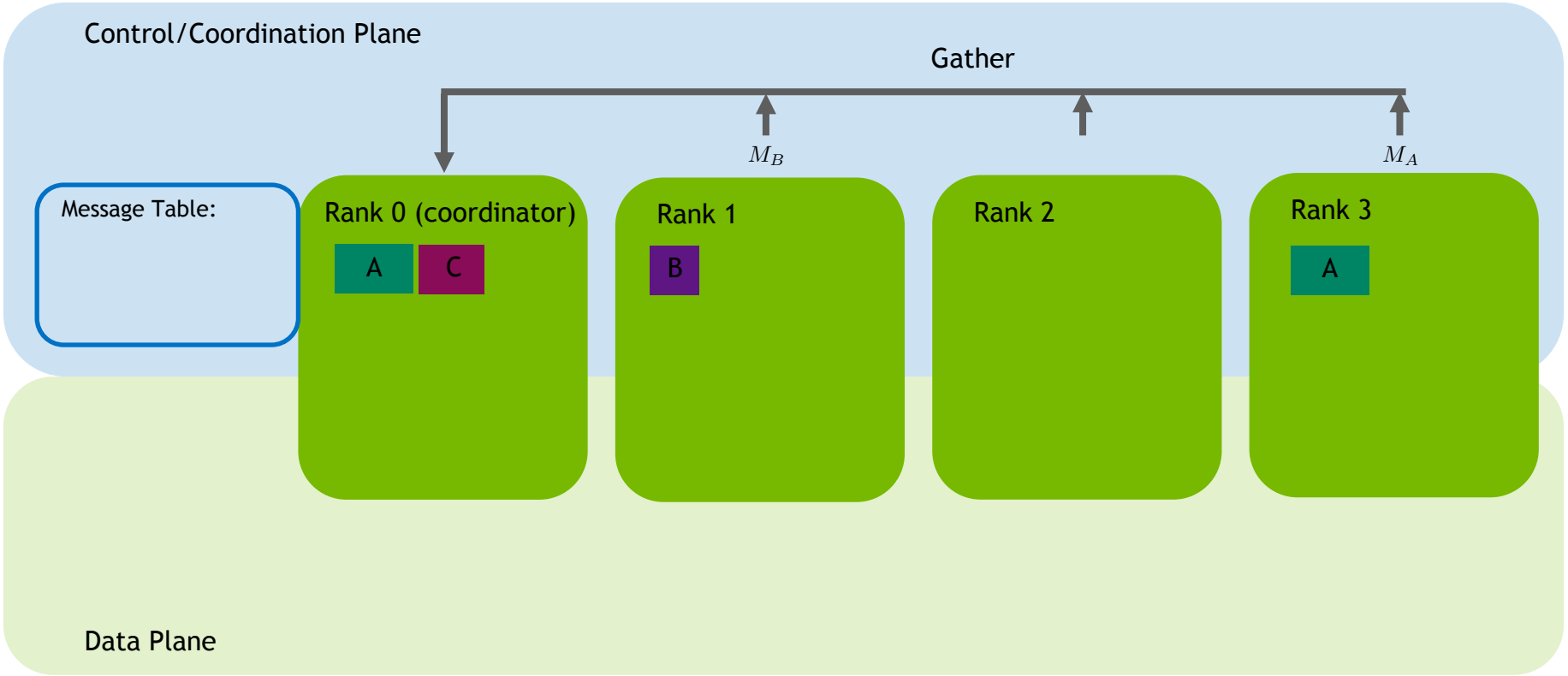
COORDINATION PROCESS

Original Design



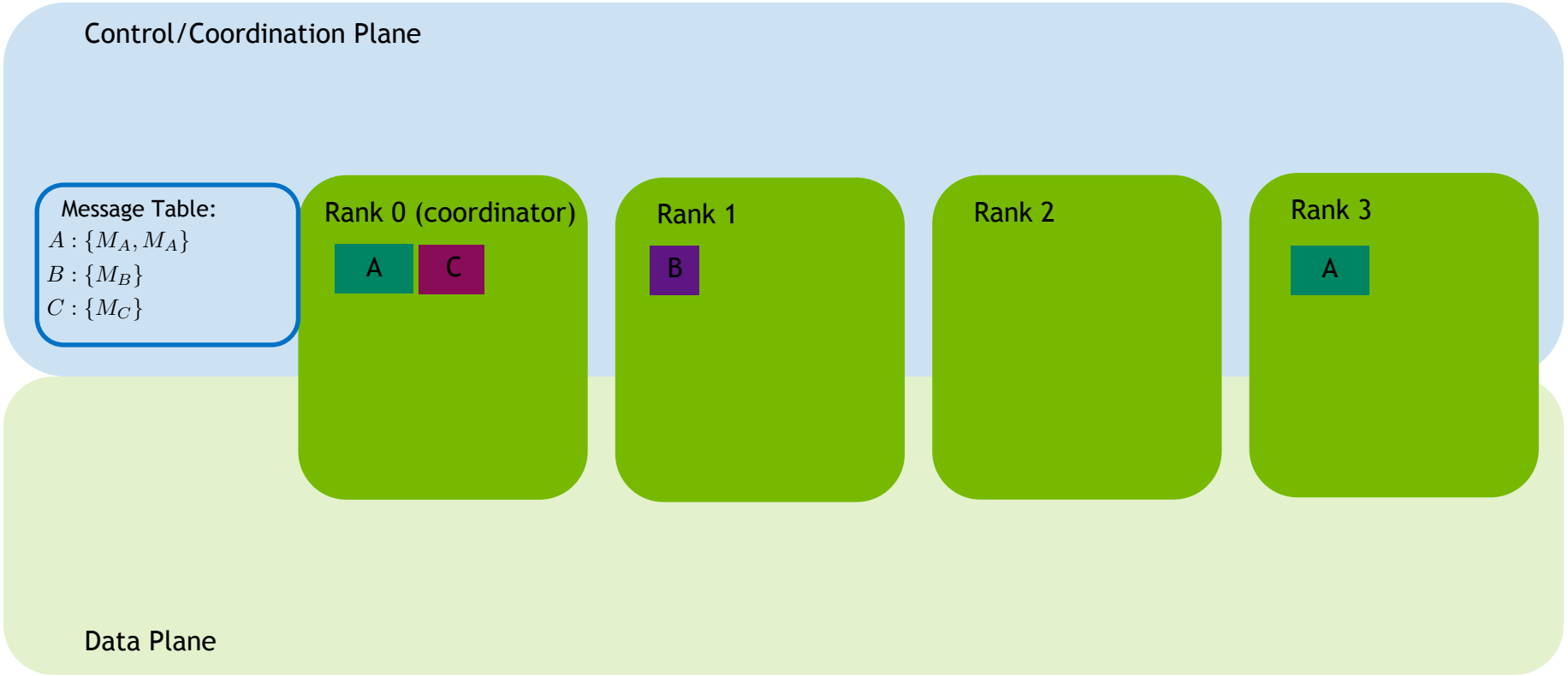
COORDINATION PROCESS

Original Design



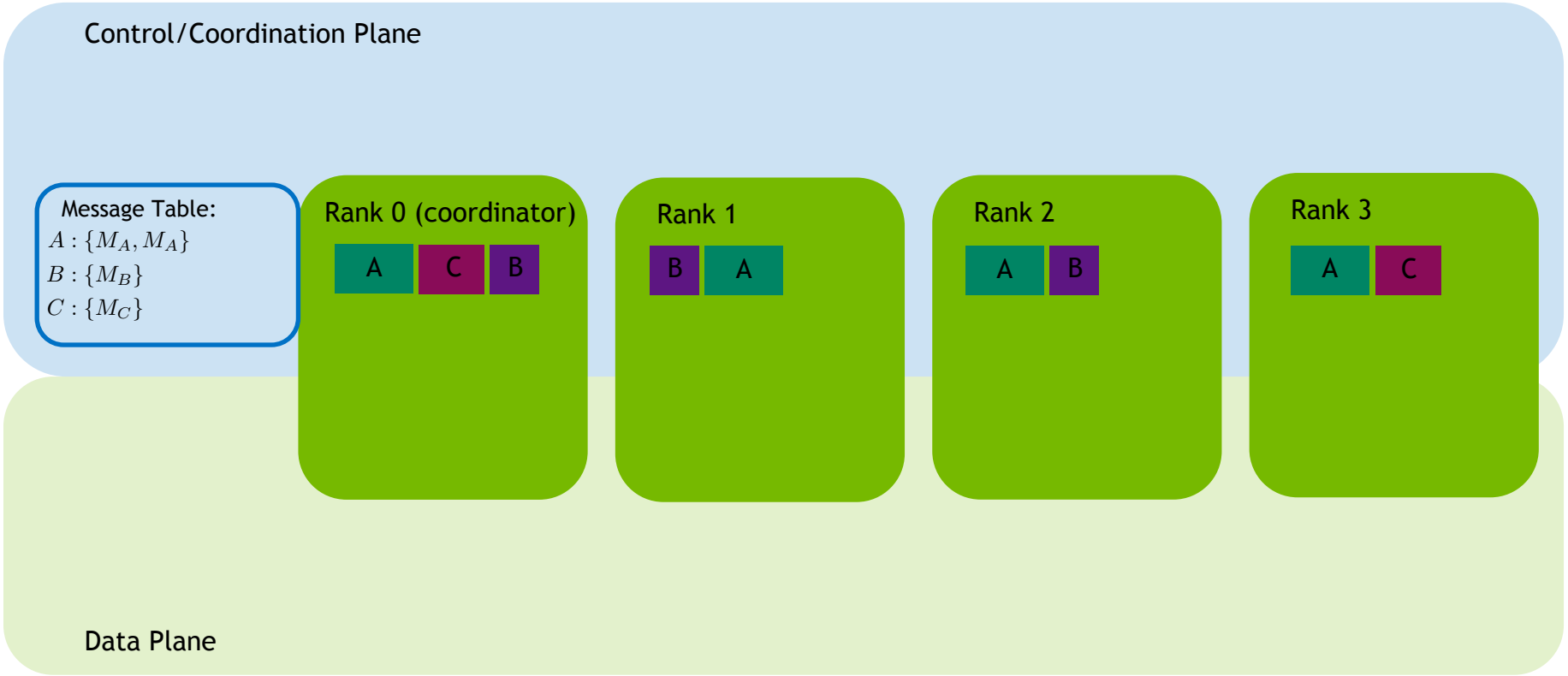
COORDINATION PROCESS

Original Design



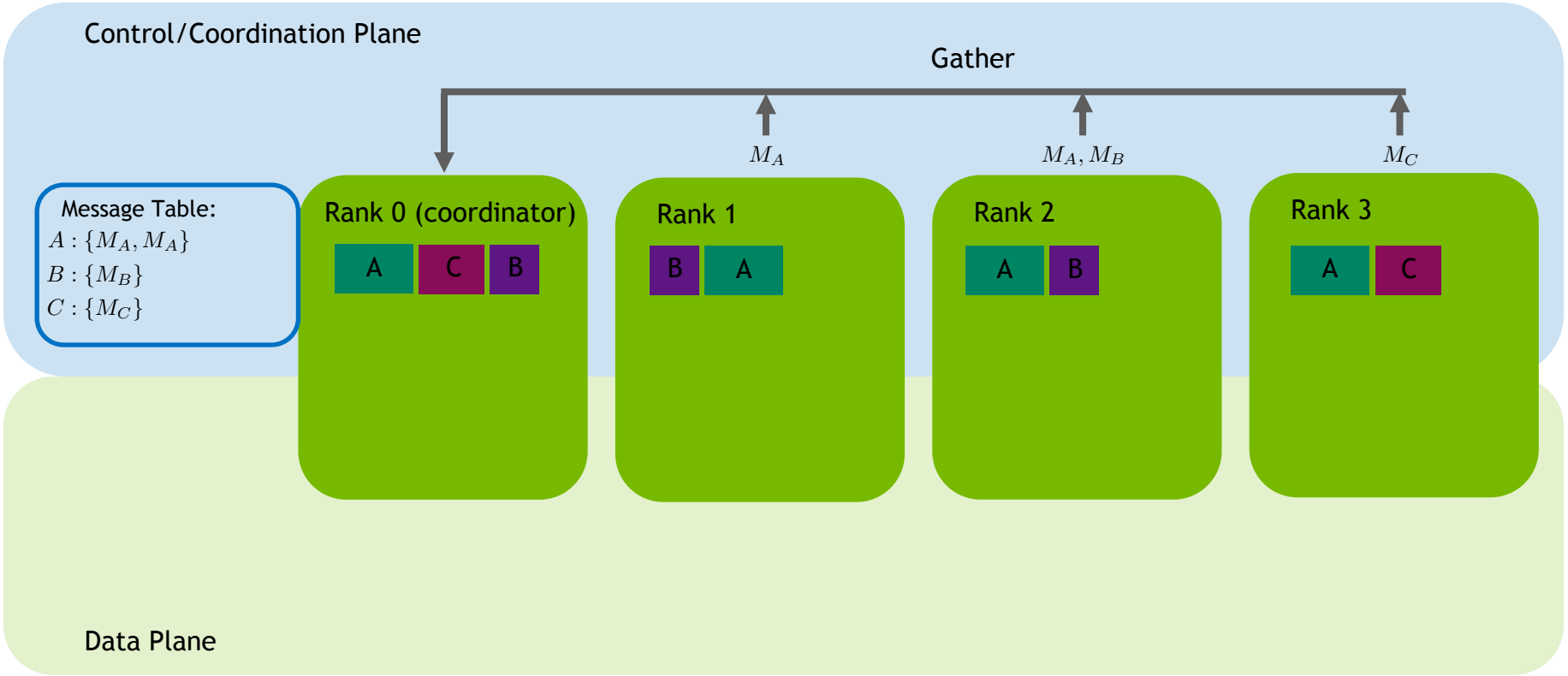
COORDINATION PROCESS

Original Design



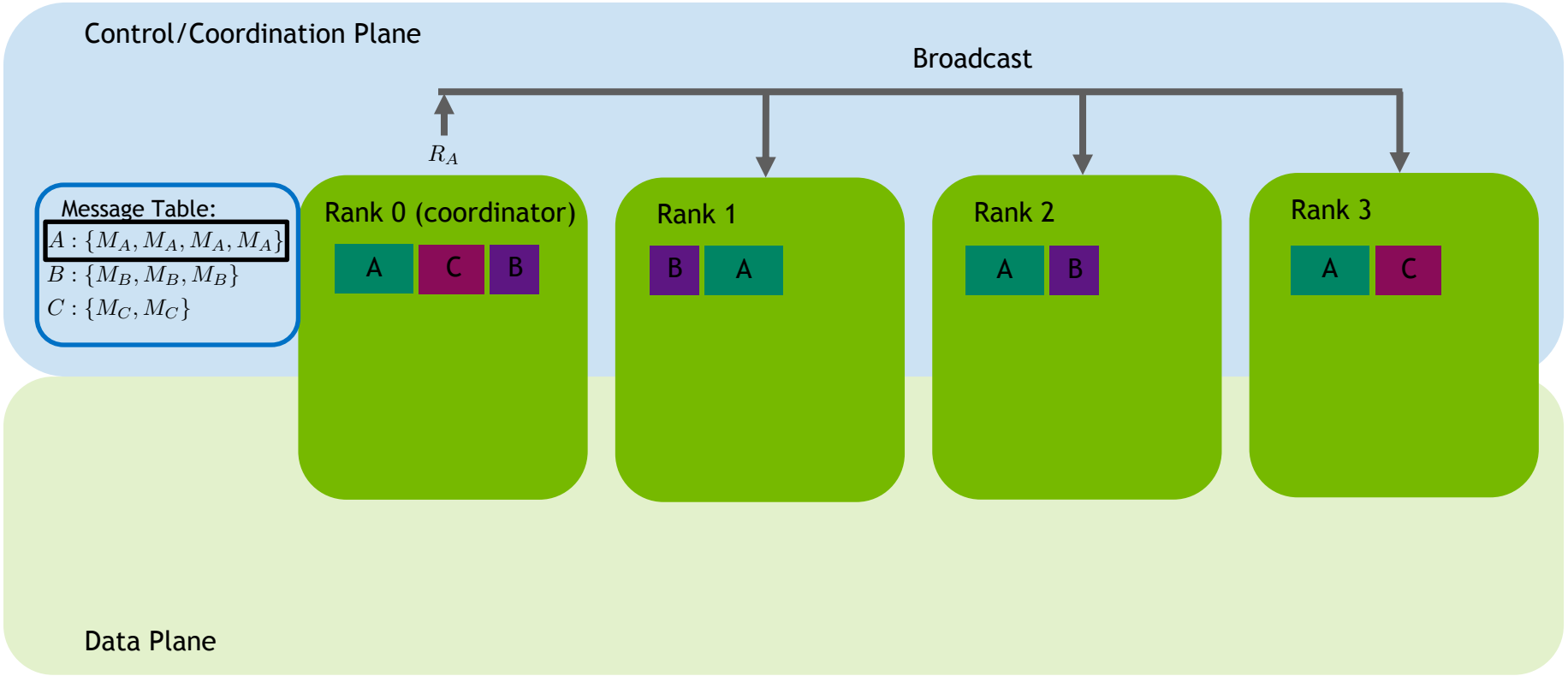
COORDINATION PROCESS

Original Design



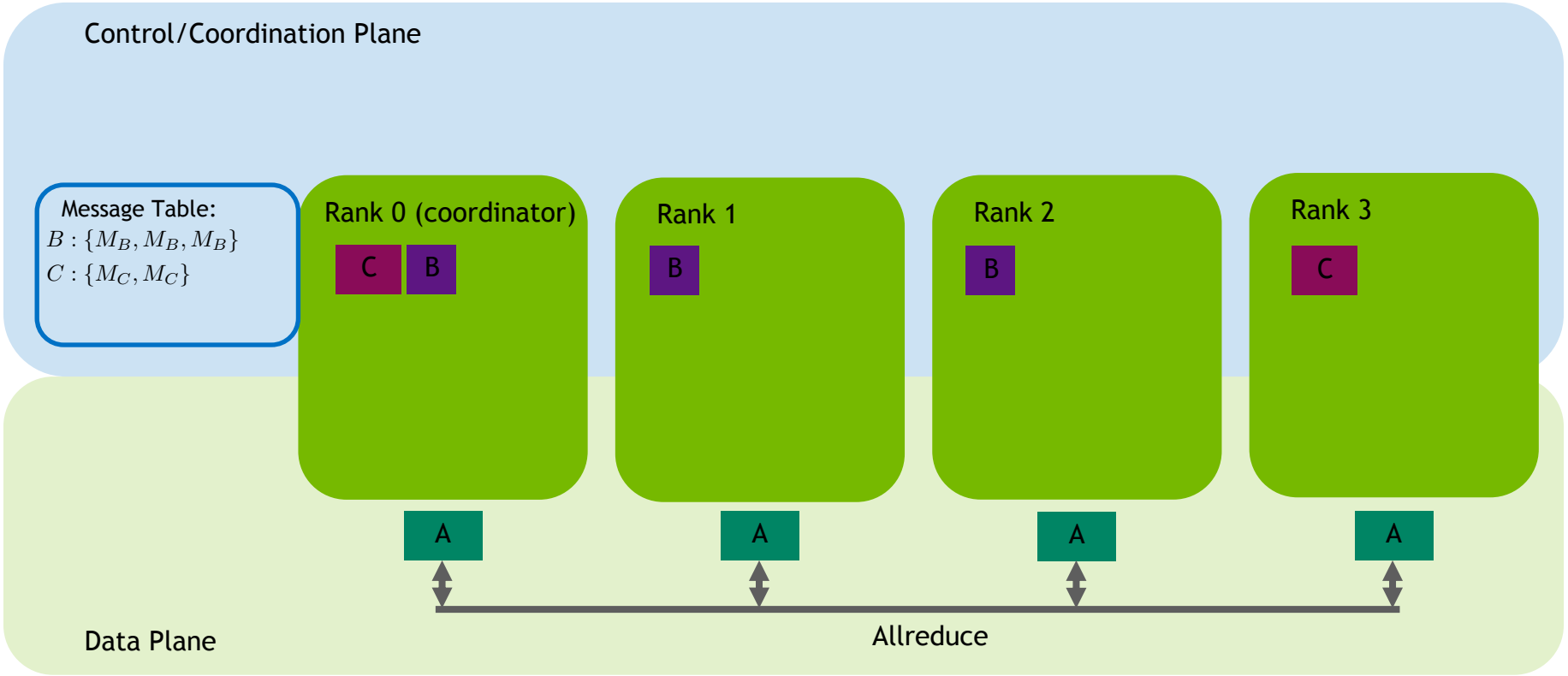
COORDINATION PROCESS

Original Design



COORDINATION PROCESS

Original Design



COORDINATION PROCESS

Scalability Issues

Relying on a single coordinator rank is a major bottleneck, especially at large scale!

These limitations were discussed in *Exascale deep learning for climate analytics* by Kurth et al. at SC18:

- Full-scale DL training on the Summit supercomputer, achieved a peak compute of 1 Exaop.
- Reported control-plane scaling issues in Horovod and implemented a fix ← never upstreamed due to lack of generality.
- Performance scaling required 1 iteration “lag” ← training wasn’t fully synchronous

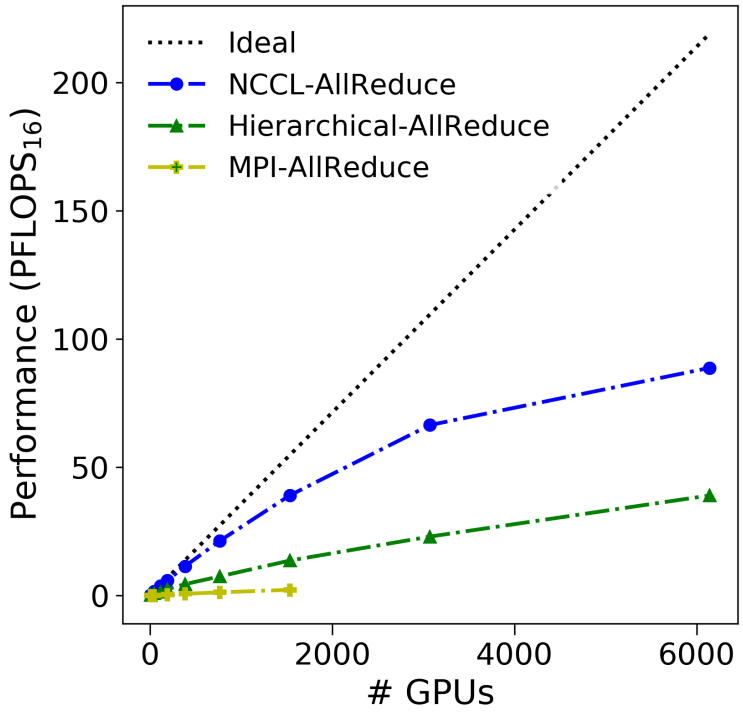
COORDINATION PROCESS

Scalability Issues

This is a scaling plot for the STEMDL model of interest in this study. At the start of this development, none of the data plane options were found to scale well, indicating that **control plane issues were still unresolved in Horovod**.

Our goals:

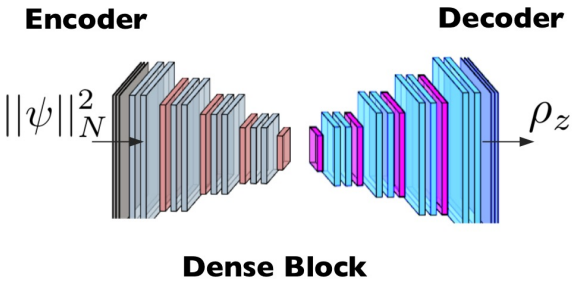
- Improve scaling performance of Horovod and contribute changes to library
- Scale training of STEMDL model to full Summit supercomputer and achieve high performance with truly synchronous training.

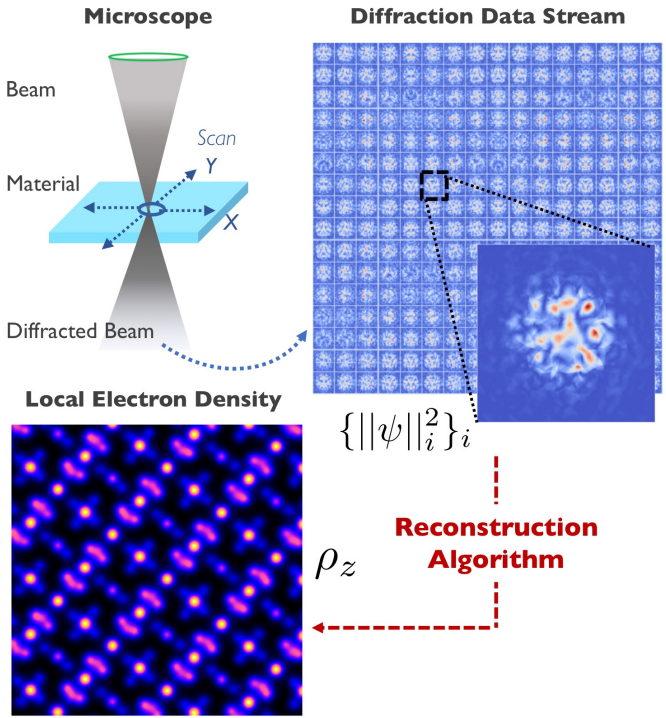


STEMDL MODEL

Image Reconstruction in Electron Microscopy

- STEMDL model attempts to solve inverse problem in EM, obtaining local electron density from diffraction data from microscope
- Implemented using an encoder/decoder model based on FC-DenseNet
- Model size of 220M trainable parameters





IMPROVEMENT 1: “RESPONSE” CACHING

Background

Observation: The set of tensors requiring communication is typically fixed during training.

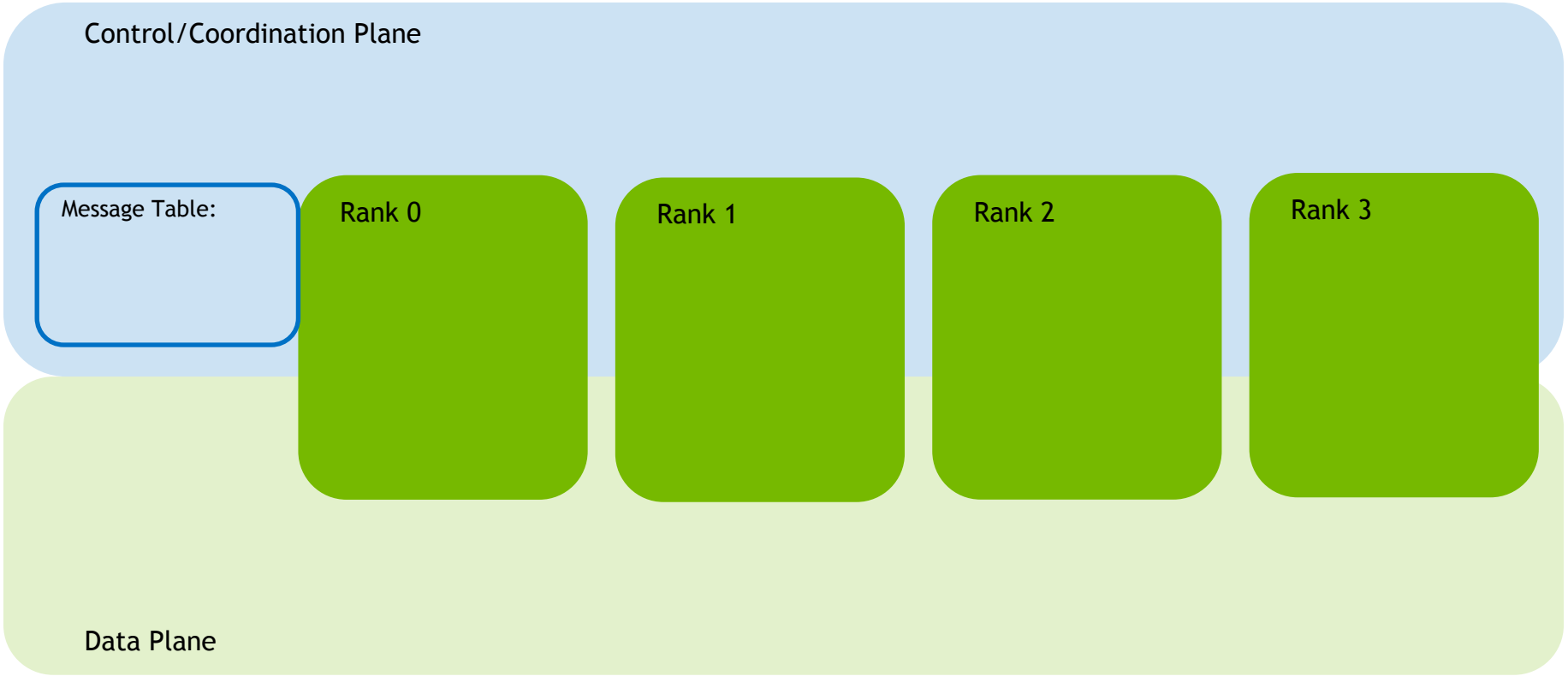
- Existing design does not take advantage of this and redundantly communicates/processes control metadata via coordinator redundantly every step.

Solution: use caching to bypass redundant control communication

- Store initial response received by the coordinator on all workers and reuse for subsequent iterations
- Construction of cache is kept globally consistent and provides enumeration of observed tensors ← enables light-weight communication using a *bitvector* to intersect worker tensor lists to determine readiness
- Readiness decisions now made collectively, rather than relying on a single coordinator rank!

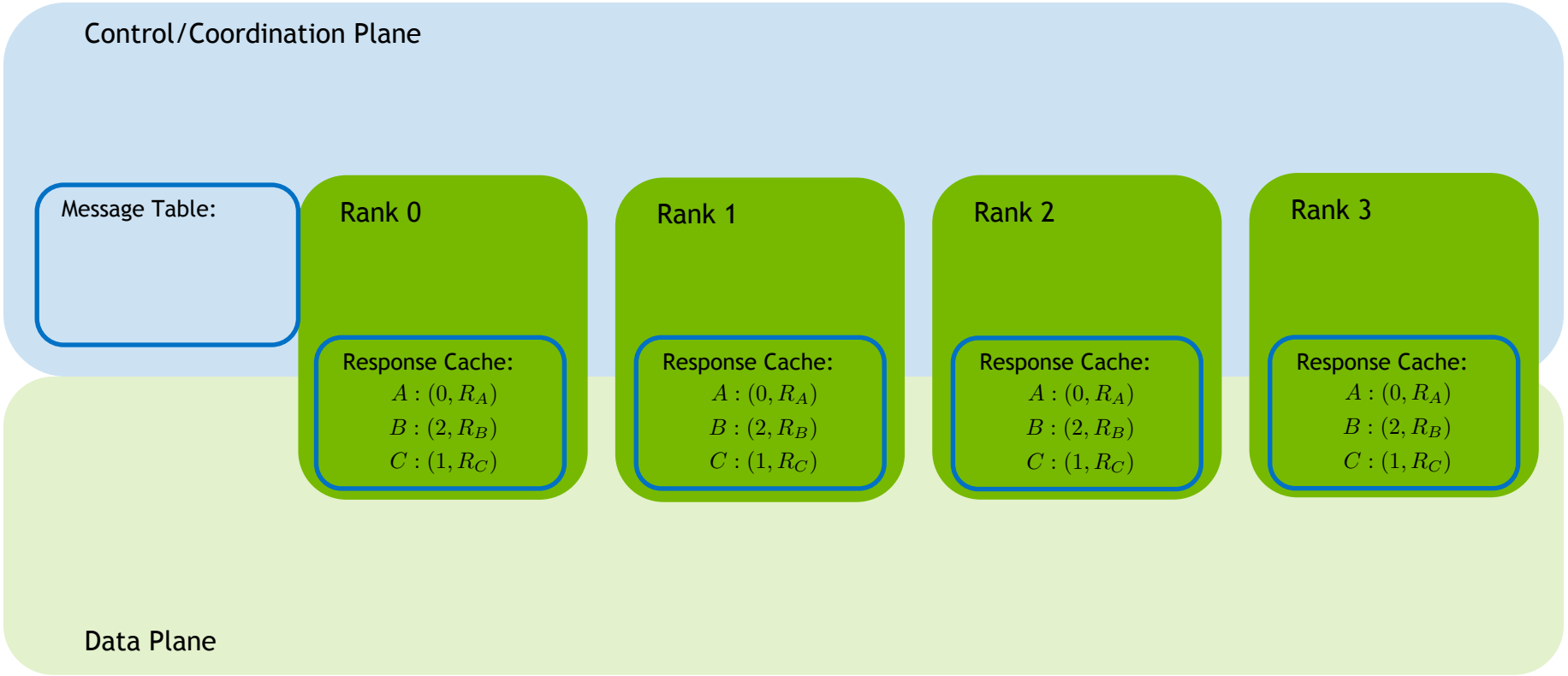
COORDINATION PROCESS

Improved Design with Caching



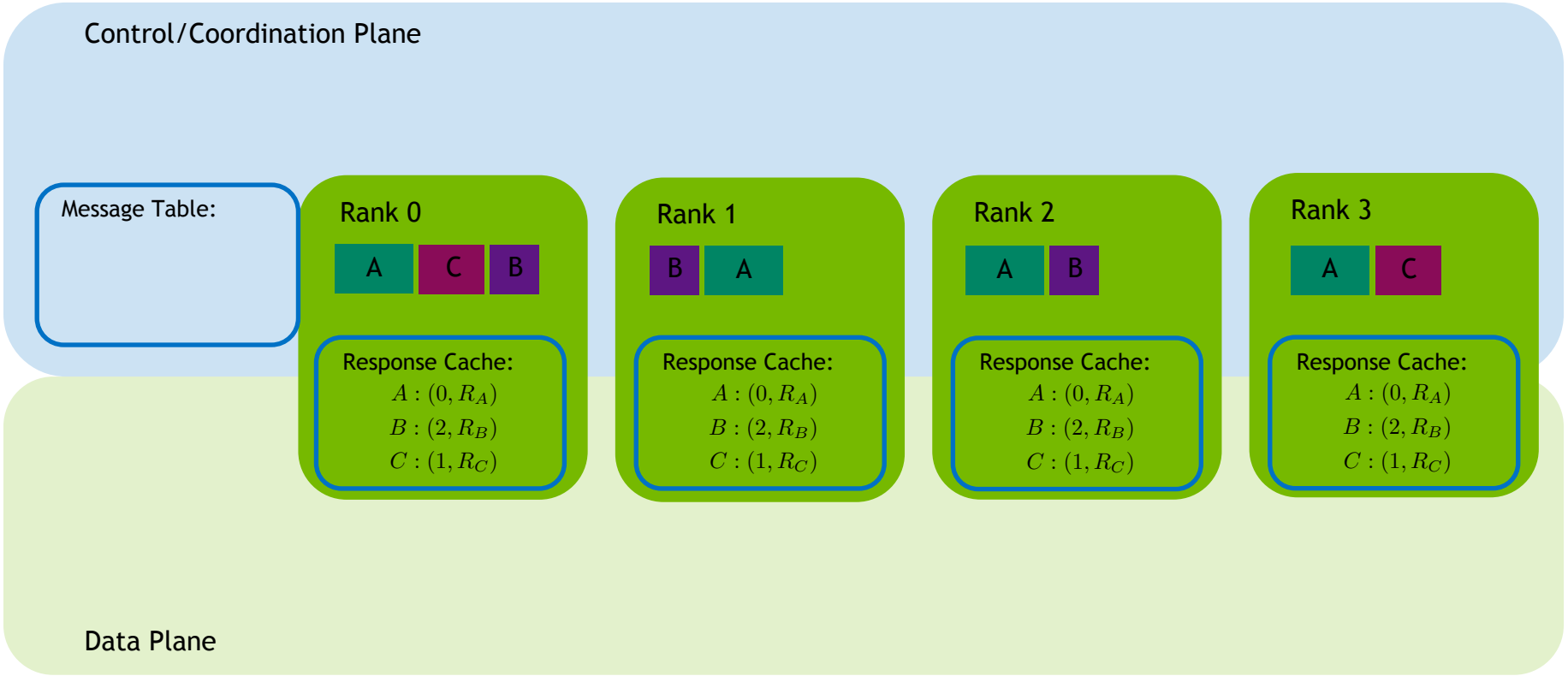
COORDINATION PROCESS

Improved Design with Caching



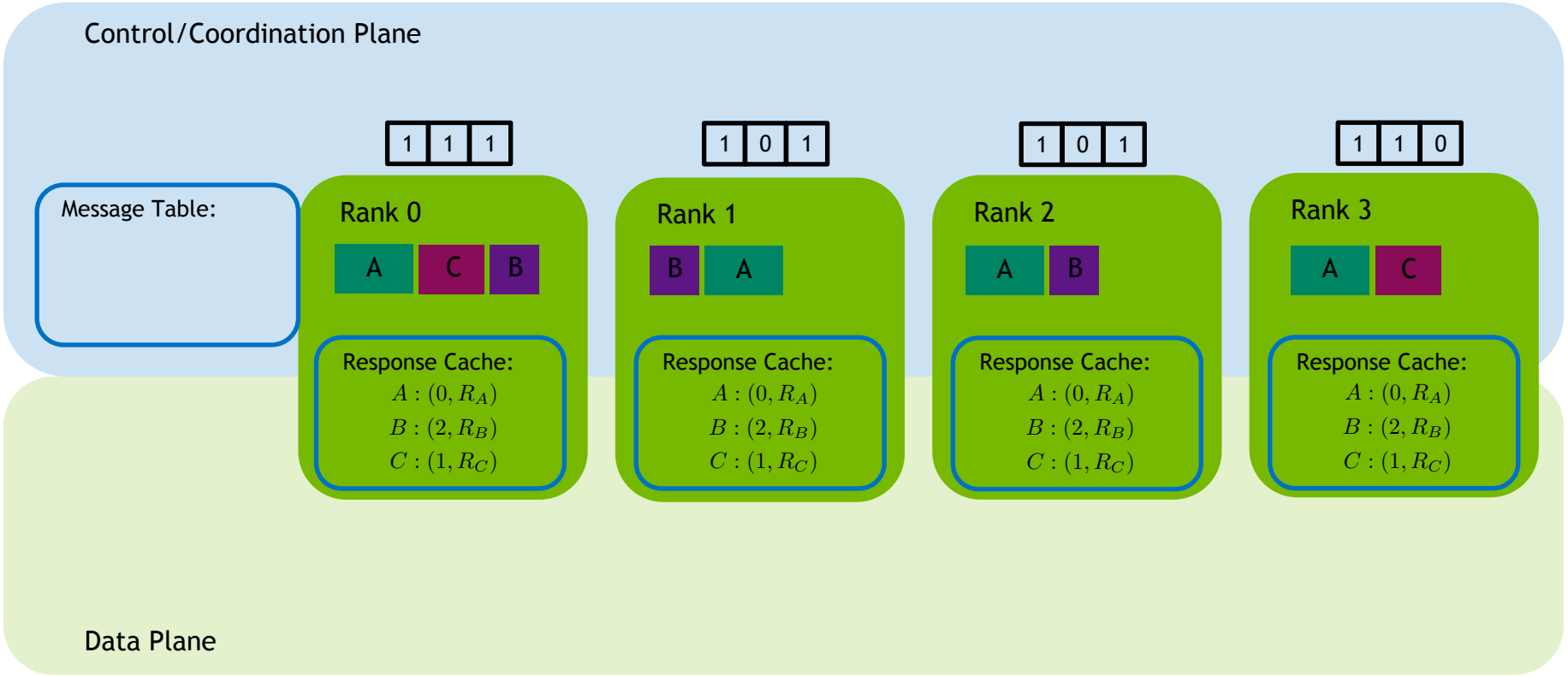
COORDINATION PROCESS

Improved Design with Caching



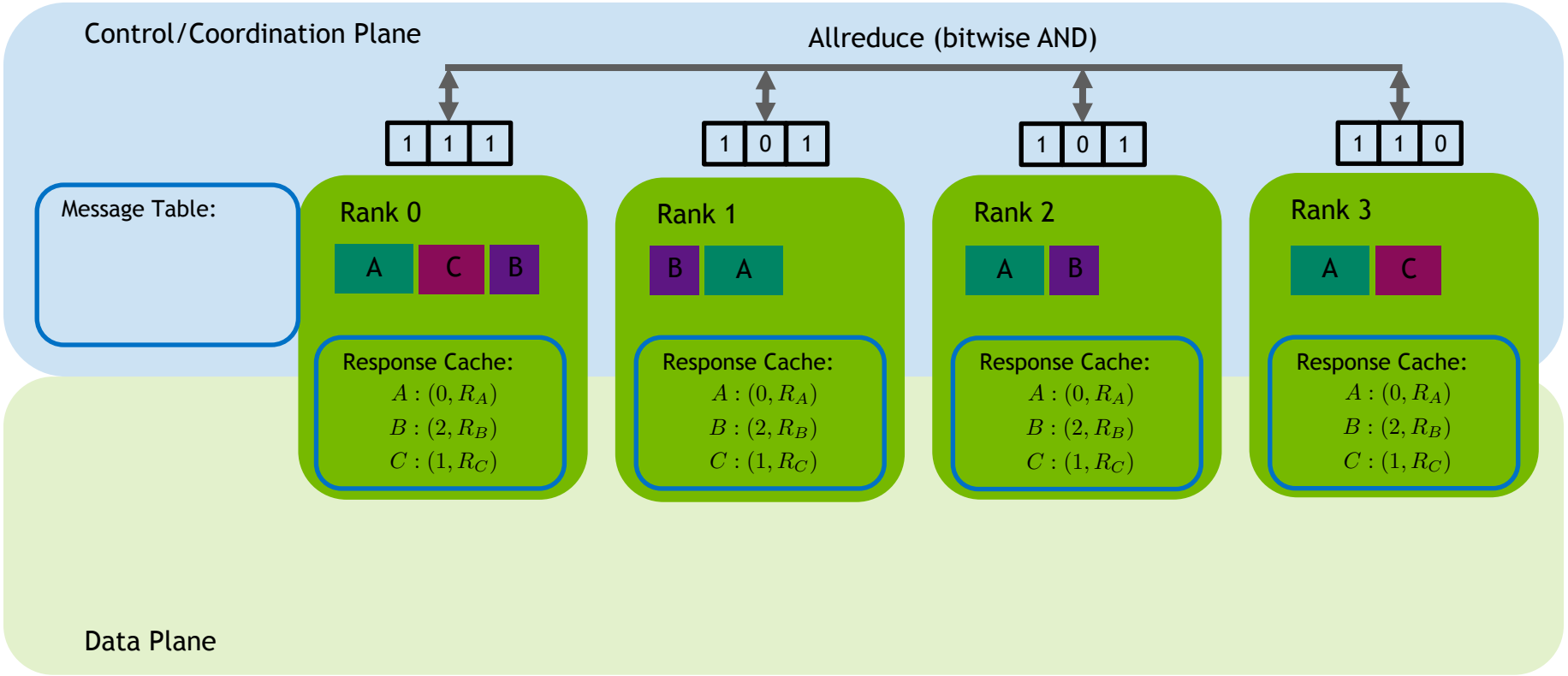
COORDINATION PROCESS

Improved Design with Caching



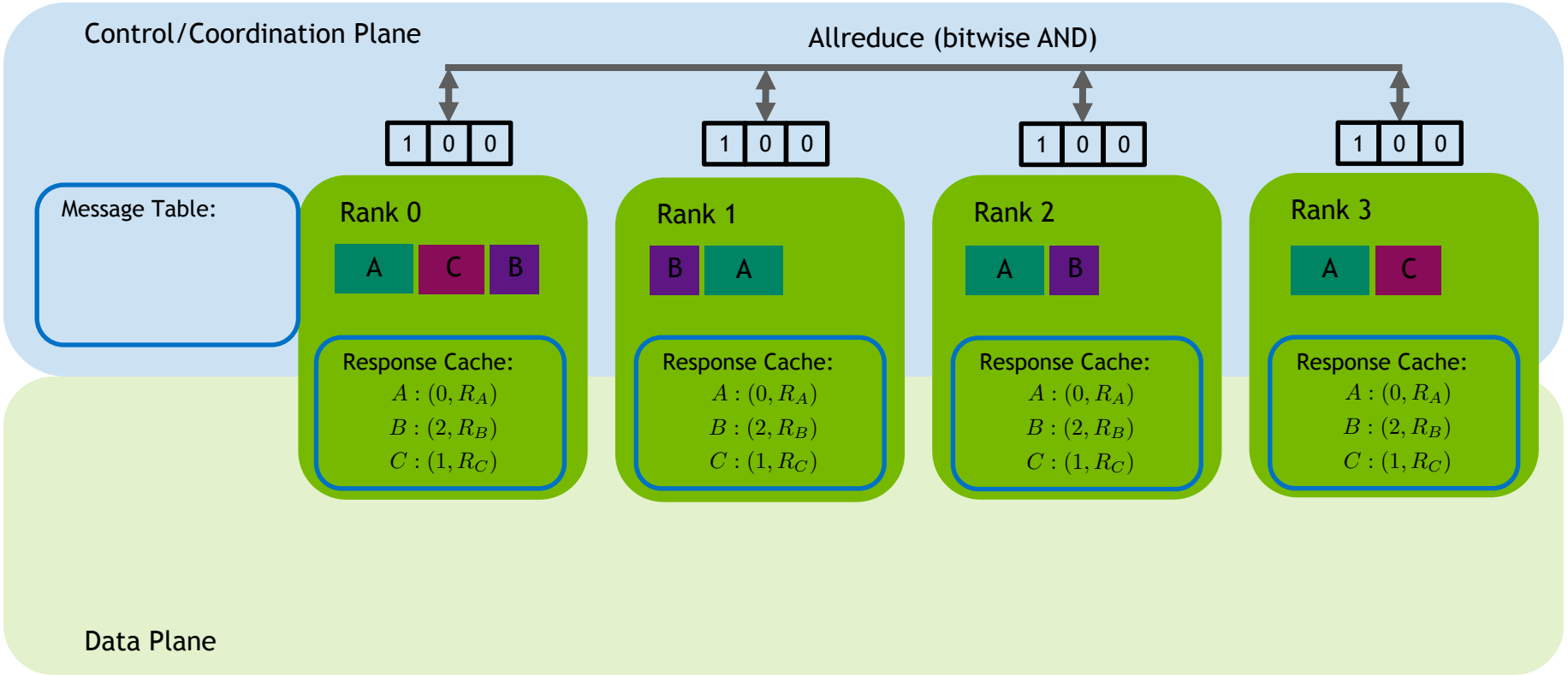
COORDINATION PROCESS

Improved Design with Caching



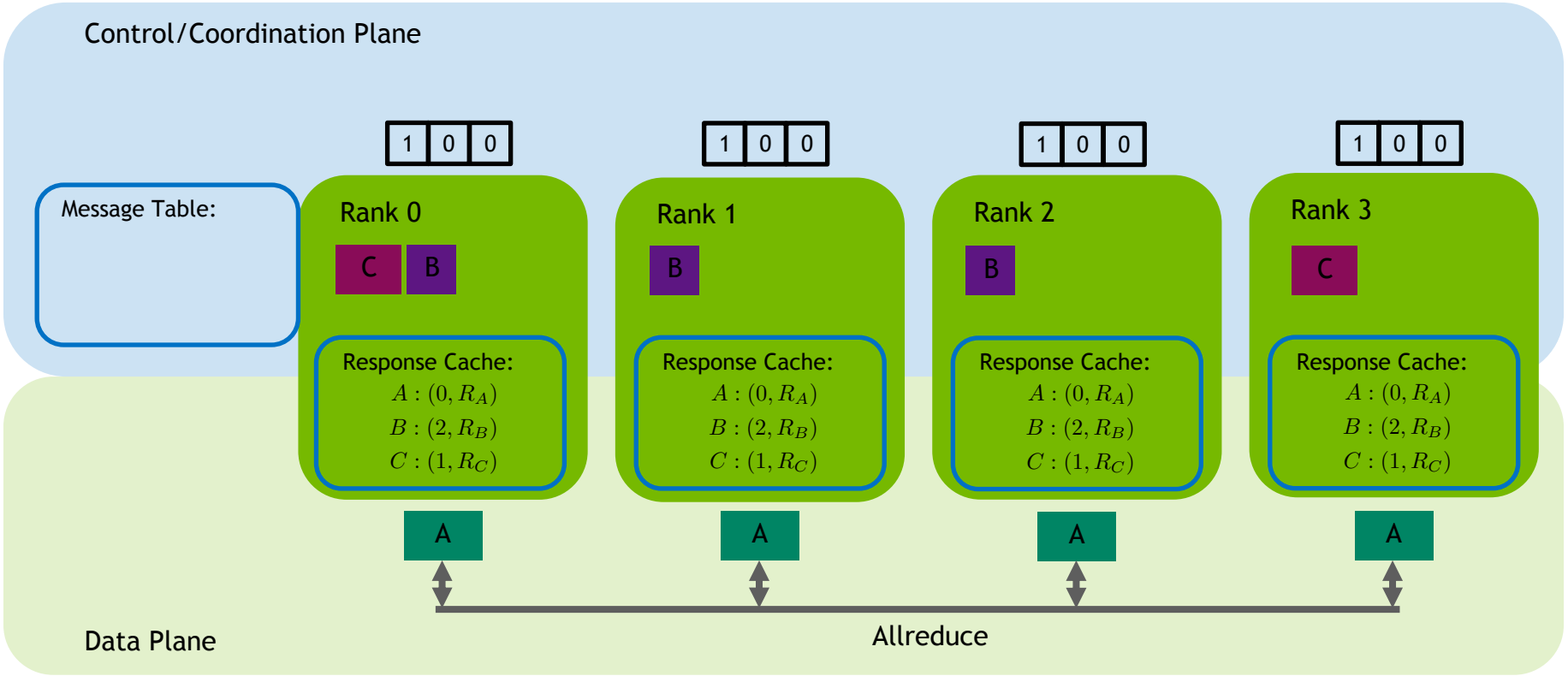
COORDINATION PROCESS

Improved Design with Caching



COORDINATION PROCESS

Improved Design with Caching



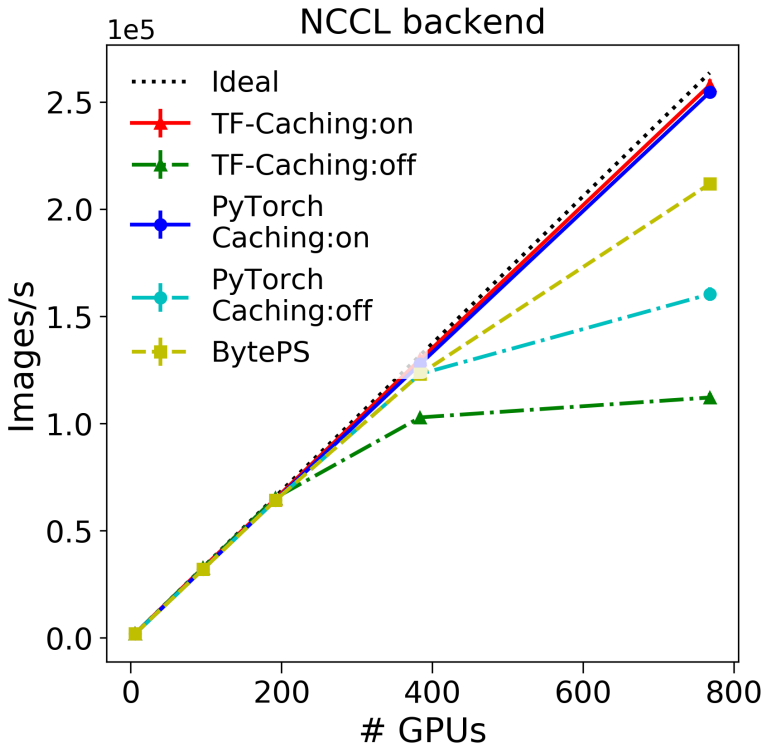
IMPACT OF CACHING

Scalability improvements

Scaling results up to ~800 GPUs on Summit on RN50 model:

- BS = 64 in FP32, simple problem to highlight impact of change

Without caching improvement, scaling drops off rapidly beyond 400 GPUs while with caching improvement, near ideal scaling is maintained.



IMPROVEMENT 2: ALLREDUCE GROUPING

Background

Observation: Horovod's tensor fusion controlled dynamically via "cycle time" setting, all tensors ready during a cycle are fused greedily

- For minimum latency, want to cycle time as low as possible to run coordination logic more often
- As a side effect, this can cause many unfused, or small buffers, to be used for collectives, which can be inefficient.

Solution: Implement grouping feature in Horovod, to explicitly control tensor fusion

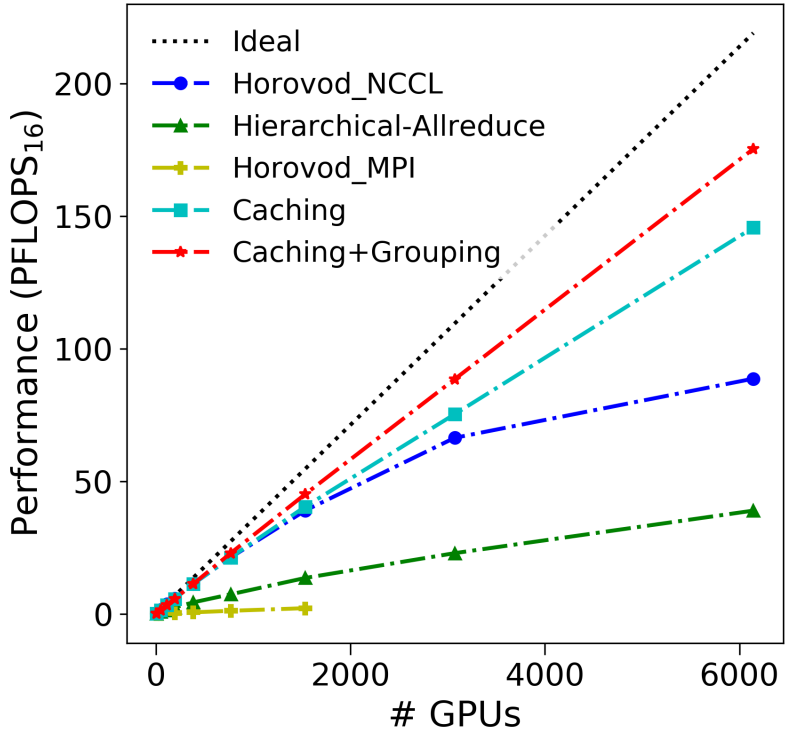
- Enables running with a low cycle time for minimum latency, while ensuring buffer sizes for collectives remain large for more efficient network utilization

IMPACT OF GROUPING

Scalability improvements

Scaling results up to ~6000 GPUs on Summit of STEMDL model.

With both caching and grouping, scaling of Horovod is significantly improved over baseline implementations.

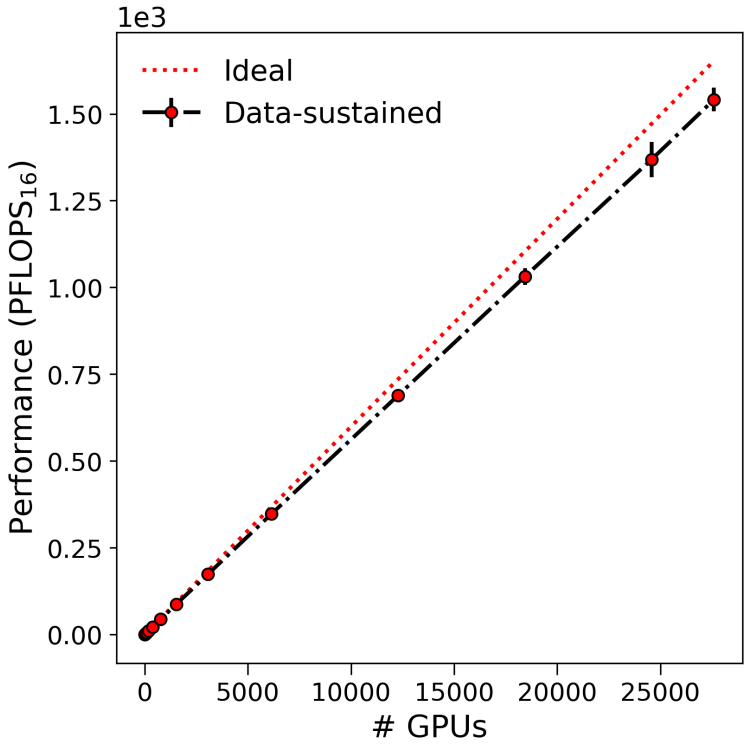


REACHING THE SUMMIT

Running STEMDL workload on full Summit system

With Horovod improvements, we were able to run the STEMDL workload on the full Summit system (27,600 GPUs):

- Achieved **1.54 exaops (sustained), 2.15 exaops (peak)** training performance, outperforming previous work by Kurth et al.
- Scaling achieved using fully synchronous training, with improvements upstreamed to Horovod



ACKNOWLEDGEMENTS

- Junqi Yin, Nouamane Laanait, Bing Xie, M. Todd Young, Vitalii Starchenko, Albina Borisevich, Michael Matheson (ORNL)
- Sean Treichler (NVIDIA)
- Alex Sergeev (formerly at Uber, Carbon Robotics)

- Shivaram Venkataraman, our NSDI shepherd

This research was partially funded by a Lab Directed Research and Development project at Oak Ridge National Laboratory, a U.S. Department of Energy facility managed by UT-Battelle, LLC. An award of computer time was provided by the INCITE program. This research also used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725


NVIDIA