Optimizing Collectives with Large Payloads on GPU-based Supercomputers

Siddharth Singh
sidsingh@nvidia.com
NVIDIA, Inc.
Santa Clara, California, USA

Mahua Singh
s.mahua@iitg.ac.in
Dept. of Computer Science and Engg.
Indian Institute of Technology
Guwahati, India

Keshav Pradeep
keshprad@umd.edu
Department of Computer Science
University of Maryland
College Park, Maryland, USA

Abhinav Bhatele
bhatele@cs.umd.edu
Department of Computer Science
University of Maryland
College Park, Maryland, USA

Abstract

We evaluate the current state of collective communication on GPU-based supercomputers for large language model (LLM) training at scale. Existing libraries such as RCCL and Cray-MPICH exhibit critical limitations on systems such as Frontier – Cray-MPICH underutilizes network and compute resources, while RCCL suffers from severe scalability issues. To address these challenges, we introduce PCCL, a communication library with highly optimized implementations of all-gather and reduce-scatter operations tailored for distributed deep learning workloads. PCCL is designed to maximally utilize all available network and compute resources and to scale efficiently to thousands of GPUs. It achieves substantial performance improvements, delivering 6–33× speedups over RCCL and 28–70× over Cray-MPICH for all-gather on 2048 GCDs of Frontier. These gains translate directly to end-to-end performance: in large-scale GPT-3-style training, PCCL provides up to 60% and 40% speedups over RCCL for 7B and 13B parameter models, respectively.

1 Collective Communication in Parallel Deep Learning

While several categories of parallelism exist in deep learning (tensor parallelism [5], pipeline parallelism [2], expert parallelism [3]), this work focus on sharded data parallelism, a widely used approach for large scale training [4, 9]. Two critical collective communication operations – *all-gather* and *reduce-scatter* – play a central role in sharded data parallelism. These operations aggregate distributed data across GPUs. In Figure 1, we plot the all-gather and reduce-scatter message sizes for three frameworks that support sharded data parallelism – FSDP [9], Deepspeed ZeRO-3 [4], and AxoNN [6]. Notice how the message sizes across these three frameworks are in the tens to hundreds of megabytes, even becoming more than a gigabyte for larger models.

2 Observed Cray-MPICH and RCCL issues on Frontier

Through benchmarking experiments, we observed the following issues:

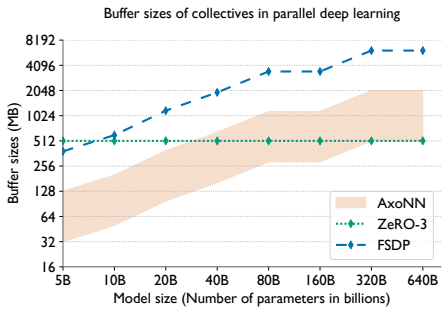

Figure 1: All-gather and reduce-scatter message sizes for various deep learning frameworks and transformer model sizes.

- Cray MPICH routes all network traffic through a single NIC, resulting in severe underutilization of the available network bandwidth.
- Cray MPICH's CPU-based reduction operations in reduce-scatter introduce significant overhead, which in combination with the NIC underutilization issue, results in a 10-15x performance gap compared to RCCL.
- Both Cray MPICH and RCCL rely solely on the ring algorithm for all-gather and reduce-scatter, leading to poor scaling in latency-bound scenarios as shown in Figure 3.

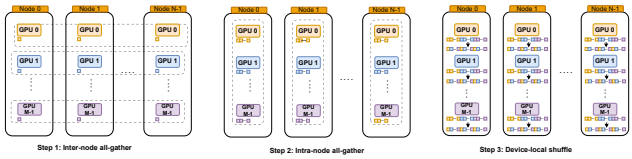

Figure 2: Diagram showing our hierarchical (two-level) implementation to dissolve an all-gather operation on a GPU-based cluster with N nodes and M GPUs per node.

3 Optimizing All-gathers and Reduce-scatters

Our optimized implementations of all-gather and reduce-scatter follow a two-level hierarchical design to mitigate the NIC underutilization issue.

The hierarchical communication proceeds in three steps detailed in Figure 2. Step 1 (inter-node all-gather): all inter-node sub-communicators perform all-gather concurrently, so each GPU receives data from its peers on all other nodes. Step 2 (intra-node all-gather): within each node, GPUs exchange data to assemble the complete result, though in an incorrect order. Step 3 (device-local shuffle): each GPU locally rearranges its data using a transpose kernel to obtain the correctly ordered output. Reduce-scatter is implemented similarly, but begins with the intra-node phase followed by the inter-node phase.

An important aspect of our design is that it schedules all of the all-gather operations in Step 1 of Figure 2 concurrently on all of the inter-node sub-communicators. We leverage this fact to utilize all NICs on a node concurrently. A Frontier node has four NICs, each connected to two GCDs. In our implementation, we ensure that each GCD exclusively sends and receives traffic to and from its corresponding NIC (e.g. - GCDs 0 and 1 to NIC 0, GCDs 2 and 3 to NIC 1, and so on), thus ensuring uniform traffic across NICs.

3.1 Choice of Communication Libraries for Each Level of the Hierarchy

We chose RCCL for intra-node communication since it is highly optimized for GPU-to-GPU intra-node communication, leveraging shared memory, PCIe, and Infinity fabric. For inter-node communication, Cray-MPICH is chosen primarily for reliability, as RCCL has been reported to be unstable and prone to crashing at scale.

3.2 Choice of Algorithms for Inter-Node Communication

Our choice of communication algorithms for each level of the hierarchy is driven by performance considerations and the limitations of available libraries.

Since RCCL only supports the ring algorithm for intra-node collectives, we adopt this as our intra-node communication strategy. Fortunately, ring is well-suited for this context, as the small number of GCDs within a node (eight) ensures that ring can effectively saturate the available bandwidth.

With thousands of GPUs participating in the collective, latency concerns become critical in the inter-node phase. Cray-MPICH (used for inter-node communication) offers only the ring algorithm by default. However, ring algorithms's linear scaling in latency with respect to processes count makes it suboptimal at large-scale.

To address this, we implement alternative algorithms with improved scaling properties. Specifically, we utilize recursive doubling for all-gather and recursive halving for reduce-scatter [7]. These algorithms offer logarithmic latency terms enabling significantly better performance as the number of GPUs increases.

4 Experimental Setup

We conduct our experiments on the Frontier supercomputer at Oak Ridge National Laboratory.

We benchmark PCCL on a range of message sizes from 16MB to 1GB. For reduce-scatter, this range represents the size of the input buffer on each GPU, while for all-gather, it corresponds to the size of the output buffer on each GPU. For each message size, we measure performance across 32 to 2048 GCDs (4 to 256 nodes) on Frontier. We use HIP and CUDA event timers to measure the runtime of collective operations.

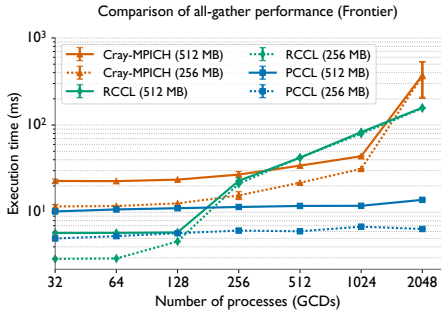

Figure 3: Performance comparison of all-gather using Cray MPICH, RCCL, and PCCL, for different per-process output buffer sizes (ideal scaling behavior is flat horizontal line)

To evaluate the practical benefits of PCCL, we measure the end-to-end training performance of large language models using DeepSpeed ZeRO-3 [4], a widely adopted parallel deep learning framework. We perform strong scaling experiments on 7B and 13B parameter GPT-style transformer models [1] using model hyperparameters from Zhang et al. [8].

Strong scaling experiments are performed between 128 and 2048 GCDs. We run 10 training batches across three trials and compute the average throughput over the last 8 batches in each run to minimize warm-up effects.

5 Results

Figure 3 shows the performance of all-gather operations on Frontier using PCCL and other communication libraries. We evaluate two sets of output buffer sizes: 64 and 128 MB (top plot), and 256 and 512 MB (bottom plot). For each configuration, we scale the number of GCDs from 32 to 2048. Since the output buffer size per GPU remains fixed, the ideal performance curve for each buffer size is a flat horizontal line, indicating perfect scaling.

While RCCL and MPICH fall short of this ideal, PCCL's all-gather performance maintains nearly flat scaling trends across message sizes and GCD counts. We observe similar trends for reduce-scatter.

Figure 4 shows the speedups of PCCL over RCCL for all-gather operation on Frontier across a range of output buffer sizes and process counts. PCCL delivers substantial gains in latency-bound scenarios, achieving speedups of 30x over RCCL's all-gather for 16-64MB message sizes at 2048 GCDs. In bandwidth bound scenarios, PCCL underperforms RCCL, delivering speedups around 0.52x over RCCL for 1024 MB buffer at 32 GCDs. We observe similar results for reduce-scatter.

Figure 5 presents the batch times for strong scaling GPT-3-style transformer training on Frontier using the DeepSpeed ZeRO-3

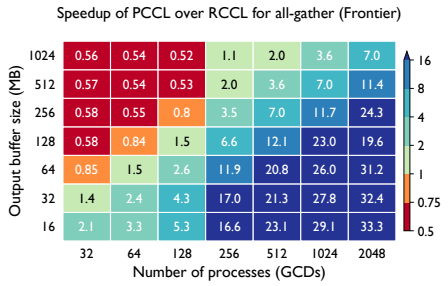

Figure 4: Speedups from using PCCL over RCCL for all-gather.

framework [4]. For end-to-end DeepSpeed ZeRO-3 training on Frontier, PCCL and RCCL perform similarly at smaller scales (128-256 GCDs). At larger scales, PCCL shows significant improvements in batch time compared to RCCL. At 2048 GCDs, PCCL reduces batch time by 72% (7B model) and 79% (13B model).

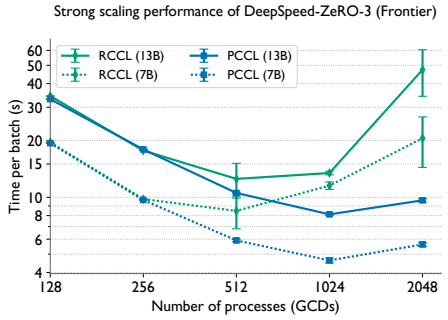

Figure 5: Strong scaling performance of Deepspeed ZeRO-3 using RCCL and PCCL, on Frontier for two model sizes: GPT-3 7B and GPT-3 13B.

6 Conclusion

In this work, we developed PCCL, a new communication library with highly optimized implementations of all-gather and reduce-scatter operations. PCCL leverages several optimizations designed to alleviate the performance bottlenecks of Cray-MPICH, RCCL and NCCL, highlighted in this work. PCCL achieves substantial performance improvements, delivering 6–33× speedups over RCCL and 28–70× over Cray-MPICH for all-gather on 2048 GCDs of Frontier. These gains translate directly to end-to-end performance: in large-scale GPT-3-style training, PCCL provides up to 60% and 40% speedups over RCCL for 7B and 13B parameter models on Frontier, respectively.

Acknowledgments

This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

References

[1] Tom B. Brown et al. 2020. Language Models are Few-Shot Learners. *CoRR* abs/2005.14165 (2020). arXiv:2005.14165 https://arxiv.org/abs/2005.14165

[2] Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, Yonghui Wu, and zhifeng Chen. 2019. GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism. In *Advances in Neural Information Processing Systems*, Vol. 32. Curran Associates, Inc.

[3] Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He. 2022. DeepSpeed-MoE: Advancing Mixture-of-Experts Inference and Training to Power Next-Generation AI Scale. doi:10.48550/ARXIV.2201.05596

[4] Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. ZeRO: Memory Optimizations toward Training Trillion Parameter Models. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (Atlanta, Georgia) *(SC '20)*. IEEE Press, Article 20, 16 pages.

[5] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2020. *Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism.* Technical Report. arXiv:1909.08053 [cs.CL]

[6] Siddharth Singh, Prajwal Singhania, Aditya Ranjan, John Kirchenbauer, Jonas Geiping, Yuxin Wen, Neel Jain, Abhimanyu Hans, Manli Shu, Aditya Tomar, Tom Goldstein, and Abhinav Bhatele. 2024. Democratizing AI: Open-source Scalable LLM Training on GPU-based Supercomputers. In *Proceedings of the ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC '24).*

[7] Rajeev Thakur and William D. Gropp. 2003. Improving the Performance of Collective Operations in MPICH. In *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, Jack Dongarra, Domenico Laforenza, and Salvatore Orlando (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 257–267.

[8] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068* (2022).

[9] Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, Alban Desmaison, Can Balioglu, Pritam Damania, Bernard Nguyen, Geeta Chauhan, Yuchen Hao, Ajit Mathews, and Shen Li. 2023. PyTorch FSDP: Experiences on Scaling Fully Sharded Data Parallel. *Proc. VLDB Endow.* 16, 12 (aug 2023), 3848–3860. doi:10.14778/3611540.3611569