ncclsee: A Lightweight Profiling Tool for NCCL

Ioannis Vardas[a], Ruben Laso Rodriguez[b], and Majid Salimi Beni[a]

[a] *TU Wien: Parallel Computing Research Group*
[b] *Universität Wien: Scientific Computing Research Group*

To achieve scalable and efficient distributed deep learning, optimized GPU communication is paramount. We introduce `ncclsee`, a lightweight profiler plugin built using version 2 of NVIDIA's Collective Communication Library (NCCL) [1] profiling interface and the NVIDIA CUDA Profiling Tools Interface (CUPTI) [3]. `ncclsee` captures communication patterns in real time, offering insights into GPU communication performance. By focusing on simplicity and efficiency, `ncclsee` enables users to pinpoint and alleviate bottlenecks in distributed workloads, making it useful for debugging and optimizing large-scale AI training workflows.

NCCL is the de facto library for GPU communication with NVIDIA GPUS in deep learning frameworks such as PyTorch and TensorFlow. It delivers high performance by leveraging advanced technologies, including RDMA (Remote Direct Memory Access) and GPUDirect, which enable direct GPU-to-GPU data transfers, over interconnects such as PCIe, Infiniband and NVLink, with minimal CPU involvement. To optimize collective operations like AllReduce, Broadcast, and AllGather, NCCL automatically selects algorithms such as the Ring or Tree, depending on message size, topology, and buffer size characteristics. Once the parameters are selected, NCCL creates a CUDA kernel to perform the collective operation on the GPUs.

Profiling NCCL behavior at scale remains challenging, tools like NVIDIA Nsight can produce highly detailed traces, but the volume of data generated makes them impractical to analyze for large GPU clusters. `ncclsee` tackles this problem by offering summary information on NCCL operations. It leverages NCCL's event callbacks, including start and stop events as well as proxy progress activity, to accurately track asynchronous operations. Because `stopEvent` indicates only that a collective has been enqueued rather than completed, `ncclsee` utilizes CUPTI to measure the time of the corresponding CUDA kernel that performs the NCCL operation on the GPUs. The main challenge for developing `ncclsee` was creating an efficient interface that associates NCCL's profiling API events to the correct CUPTI's event.

`ncclsee` captures summary data for each NCCL operation based on the buffer size, presenting information in a concise format (e.g., Operation: `ncclAllReduce`, Buffer range: 128-4096 Bytes, Calls: 52, Time: 770 ms), allowing users to quickly identify communication patterns and potential bottlenecks.

Ease of use: Integrating `ncclsee` into existing workflows is straightforward. After compiling `ncclsee` producing the `libnccl-profiler.so`, users simply have to set the `NCCL_PROFILER_PLUGIN` environment variable to point to the path of `libnccl-profiler.so`. Once enabled, `ncclsee` records metrics for applications that use NCCL directly or through frameworks that depend on it, such as PyTorch and TensorFlow. `ncclsee` is actively under development, with a functional version already available on GitHub [3]

References

[1] NVIDIA Collective Communication Library (NCCL), `https://developer.nvidia.com/nccl`.

[2] NVIDIA CUDA Profiling Tools Interface (CUPTI), `https://developer.nvidia.com/cupti`.

[3] ncclsee: A lightweight profiling tool for NCCL, `https://github.com/variemai/ncclsee`.