

A Collective Communication:-

- * Receive Buffers + Output Buffer
- * nccl → $\left. \begin{array}{l} \text{AllReduce} \\ \text{Broadcast} \\ \text{Reduce} \\ \text{AllGather} \\ \text{Reduce Scatter} \end{array} \right\} \text{API signature}$

* All Reduce = Reduce + Broadcast
= Reduce Scatter + All Gather

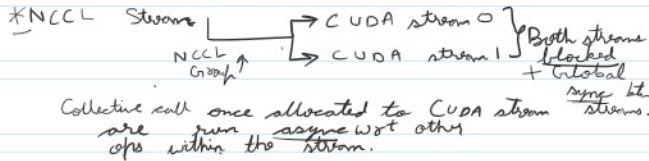
* Data Pointers or how does NCCL interface w/ memory

- 1) Accepts any CUDA pointer available on CUDA device linked to current communicator
- 2) Direct Device memory
- 3) Host memory registered to device
→ CUDA SDK → cudaHostRegister
- 4) Managed & unified memory

But it doesn't work with RDMA & GPUDirect based remote device memory.

NCCL - CHECK POINTERS = 1

Needs to be true for remote memory checks to apply.



C P2P → Calls blocking as a group but within a group they're independent.

* NCCL is not thread safe. Only 1 thread should interact w/ a communicator while others probe/poll status.

* For in-place NCCL ops, simply have send Buffer = Recv Buffer

D Copy

User buffer registration / zero-copy ops:-

- Either CUDA Graph Registration
- Local Registration

Prevents "internal" copy of data.

Must register from all ranks + sender buffers & receiver registration

* Types of user buffers →

- 1) NVLink SHARP Buffers
- 2) IB SHARP Buffers
- 3) General Buffer Registration

B Group Calls:- Merge calls into 1.

- Avoids collective launch overhead.
- 1 thread → Multiple GPUs (somehow avoids deadlock)
- Create new P2P ops out of send & recv.

* Communicated unit for rank can't be merged in the other calls into a group.

* Managing multiple GPUs from the same thread:-

- Node group semantics otherwise deadlock.

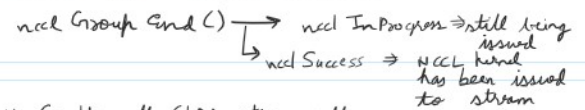
Note: 1) Init of comm should be in a group call but also the only collective op inside that group.

2) Enqueue in stream is not guaranteed inside a group call b/c by definition it will try to bundle for later call.

- Stream ops like cudaStreamSynchronize can only be called after group ends.

* Non-blocking group call

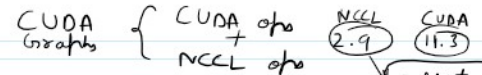
- No guarantee that group is issued to CUDA stream.



* Can't call CUDA stream calls until NCCL kernel is issued to stream.

D CUDA GRAPHS:-

* NCCL 2.9 supports NCCL ops being captured in CUDA graphs.



* Currently only CUDA ops are captured in CUDA graph.

current version is 2.27 which is much higher

```
cudaGraph_t graph;
cudaStreamBeginCapture(stream);
kernel_A<<< ..., stream >>>(...);
kernel_B<<< ..., stream >>>(...);
ncclAllreduce(..., stream);
kernel_C<<< ..., stream >>>(...);
cudaStreamEndCapture(stream, &graph);

cudaGraphExec_t instance;
cudaGraphInstantiate(&instance, graph, NULL, NULL, 0);
cudaGraphLaunch(instance, stream);
cudaStreamSynchronize(stream);
```

→ NCCL ops