[NCCL](index.html)

[2.25](https://docs.nvidia.com/deeplearning/sdk/nccl-archived/index.html)

    * [ncclGetLastError](api/comms.html#ncclgetlasterror)
    * [ncclGetErrorString](api/comms.html#ncclgeterrorstring)
    * [ncclGetVersion](api/comms.html#ncclgetversion)
    * [ncclGetUniqueId](api/comms.html#ncclgetuniqueid)
    * [ncclCommInitRank](api/comms.html#ncclcomminitrank)
    * [ncclCommInitAll](api/comms.html#ncclcomminitall)
    * [ncclCommInitRankConfig](api/comms.html#ncclcomminitrankconfig)
    * [ncclCommInitRankScalable](api/comms.html#ncclcomminitrankscalable)
    * [ncclCommSplit](api/comms.html#ncclcommsplit)
    * [ncclCommFinalize](api/comms.html#ncclcommfinalize)
    * [ncclCommDestroy](api/comms.html#ncclcommdestroy)
    * [ncclCommAbort](api/comms.html#ncclcommabort)
    * [ncclCommGetAsyncError](api/comms.html#ncclcommgetasyncerror)
    * [ncclCommCount](api/comms.html#ncclcommcount)
    * [ncclCommCuDevice](api/comms.html#ncclcommcudevice)
    * [ncclCommUserRank](api/comms.html#ncclcommuserrank)
    * [ncclCommRegister](api/comms.html#ncclcommregister)
    * [ncclCommDeregister](api/comms.html#ncclcommderegister)
    * [ncclMemAlloc](api/comms.html#ncclmemalloc)
    * [ncclMemFree](api/comms.html#ncclmemfree)
  * [Collective Communication Functions](api/colls.html)
    * [ncclAllReduce](api/colls.html#ncclallreduce)
    * [ncclBroadcast](api/colls.html#ncclbroadcast)
    * [ncclReduce](api/colls.html#ncclreduce)
    * [ncclAllGather](api/colls.html#ncclallgather)
    * [ncclReduceScatter](api/colls.html#ncclreducescatter)
  * [Group Calls](api/group.html)

* [NCCL_SOCKET_IFNAME](env.html#nccl-socket-ifname)
  * [Values accepted](env.html#values-accepted)
* [NCCL_SOCKET_FAMILY](env.html#nccl-socket-family)
  * [Values accepted](env.html#id2)
* [NCCL_SOCKET_RETRY_CNT](env.html#nccl-socket-retry-cnt)
  * [Values accepted](env.html#id3)
* [NCCL_SOCKET_RETRY_SLEEP_MSEC](env.html#nccl-socket-retry-sleep-msec)
  * [Values accepted](env.html#id4)
* [NCCL_SOCKET_NTHREADS](env.html#nccl-socket-nthreads)
  * [Values accepted](env.html#id5)
* [NCCL_NSOCKS_PERTHREAD](env.html#nccl-nsocks-perthread)
  * [Values accepted](env.html#id6)
* [NCCL_CROSS_NIC](env.html#nccl-cross-nic)
  * [Values accepted](env.html#id7)
* [NCCL_IB_HCA](env.html#nccl-ib-hca)
  * [Values accepted](env.html#id8)
* [NCCL_IB_TIMEOUT](env.html#nccl-ib-timeout)
  * [Values accepted](env.html#id9)
* [NCCL_IB_RETRY_CNT](env.html#nccl-ib-retry-cnt)
  * [Values accepted](env.html#id10)
* [NCCL_IB_GID_INDEX](env.html#nccl-ib-gid-index)
  * [Values accepted](env.html#id11)
* [NCCL_IB_ADDR_FAMILY](env.html#nccl-ib-addr-family)
  * [Values accepted](env.html#id12)
* [NCCL_IB_ADDR_RANGE](env.html#nccl-ib-addr-range)
  * [Values accepted](env.html#id13)
* [NCCL_IB_ROCE_VERSION_NUM](env.html#nccl-ib-roce-version-num)

  * [Values accepted](env.html#id14)

* [NCCL_IB_SL](env.html#nccl-ib-sl)

  * [Values accepted](env.html#id15)

* [NCCL_IB_TC](env.html#nccl-ib-tc)

  * [Values accepted](env.html#id16)

* [NCCL_IB_FIFO_TC](env.html#nccl-ib-fifo-tc)

  * [Values accepted](env.html#id17)

* [NCCL_IB_RETURN_ASYNC_EVENTS](env.html#nccl-ib-return-async-events)

  * [Values accepted](env.html#id18)

* [NCCL_OOB_NET_ENABLE](env.html#nccl-oob-net-enable)

  * [Values accepted](env.html#id19)

* [NCCL_OOB_NET_IFNAME](env.html#nccl-oob-net-ifname)

  * [Values accepted](env.html#id20)

* [NCCL_UID_STAGGER_THRESHOLD](env.html#nccl-uid-stagger-threshold)

  * [Values accepted](env.html#id21)

* [NCCL_UID_STAGGER_RATE](env.html#nccl-uid-stagger-rate)

  * [Values accepted](env.html#id22)

* [NCCL_NET](env.html#nccl-net)

  * [Values accepted](env.html#id23)

* [NCCL_NET_PLUGIN](env.html#nccl-net-plugin)

  * [Values accepted](env.html#id24)

* [NCCL_TUNER_PLUGIN](env.html#nccl-tuner-plugin)

  * [Values accepted](env.html#id25)

* [NCCL_PROFILER_PLUGIN](env.html#nccl-profiler-plugin)

  * [Values accepted](env.html#id26)

* [NCCL_IGNORE_CPU_AFFINITY](env.html#nccl-ignore-cpu-affinity)

  * [Values accepted](env.html#id27)

  * [Values accepted](env.html#id40)

* [NCCL_BUFFSIZE](env.html#nccl-buffsize)

  * [Values accepted](env.html#id41)

* [NCCL_NTHREADS](env.html#nccl-nthreads)

  * [Values accepted](env.html#id42)

* [NCCL_MAX_NCHANNELS](env.html#nccl-max-nchannels)

  * [Values accepted](env.html#id43)

* [NCCL_MIN_NCHANNELS](env.html#nccl-min-nchannels)

  * [Values accepted](env.html#id44)

* [NCCL_CHECKS_DISABLE](env.html#nccl-checks-disable)

  * [Values accepted](env.html#id45)

* [NCCL_CHECK_POINTERS](env.html#nccl-check-pointers)

  * [Values accepted](env.html#id46)

* [NCCL_LAUNCH_MODE](env.html#nccl-launch-mode)

  * [Values accepted](env.html#id47)

* [NCCL_IB_DISABLE](env.html#nccl-ib-disable)

  * [Values accepted](env.html#id48)

* [NCCL_IB_AR_THRESHOLD](env.html#nccl-ib-ar-threshold)

  * [Values accepted](env.html#id49)

* [NCCL_IB_QPS_PER_CONNECTION](env.html#nccl-ib-qps-per-connection)

  * [Values accepted](env.html#id50)

* [NCCL_IB_SPLIT_DATA_ON_QPS](env.html#nccl-ib-split-data-on-qps)

  * [Values accepted](env.html#id51)

* [NCCL_IB_CUDA_SUPPORT](env.html#nccl-ib-cuda-support)

  * [Values accepted](env.html#id52)

* [NCCL_IB_PCI_RELAXED_ORDERING](env.html#nccl-ib-pci-relaxed-ordering)

  * [Values accepted](env.html#id53)

* [NCCL_IB_ADAPTIVE_ROUTING](env.html#nccl-ib-adaptive-routing)
  * [Values accepted](env.html#id54)
* [NCCL_IB_ECE_ENABLE](env.html#nccl-ib-ece-enable)
  * [Values accepted](env.html#id55)
* [NCCL_MEM_SYNC_DOMAIN](env.html#nccl-mem-sync-domain)
  * [Values accepted](env.html#id56)
* [NCCL_CUMEM_ENABLE](env.html#nccl-cumem-enable)
  * [Values accepted](env.html#id57)
* [NCCL_CUMEM_HOST_ENABLE](env.html#nccl-cumem-host-enable)
  * [Values accepted](env.html#id58)
* [NCCL_NET_GDR_LEVEL (formerly NCCL_IB_GDR_LEVEL)](env.html#nccl-net-gdr-level-formerly-nccl-ib-gdr-level)
  * [Values accepted](env.html#id59)
  * [Integer Values (Legacy)](env.html#id60)
* [NCCL_NET_GDR_READ](env.html#nccl-net-gdr-read)
  * [Values accepted](env.html#id61)
* [NCCL_NET_SHARED_BUFFERS](env.html#nccl-net-shared-buffers)
  * [Value accepted](env.html#id62)
* [NCCL_NET_SHARED_COMMS](env.html#nccl-net-shared-comms)
  * [Value accepted](env.html#id63)
* [NCCL_SINGLE_RING_THRESHOLD](env.html#nccl-single-ring-threshold)
  * [Values accepted](env.html#id64)
* [NCCL_LL_THRESHOLD](env.html#nccl-ll-threshold)
  * [Values accepted](env.html#id65)
* [NCCL_TREE_THRESHOLD](env.html#nccl-tree-threshold)
  * [Values accepted](env.html#id66)
* [NCCL_ALGO](env.html#nccl-algo)

  * [Values accepted](env.html#id67)

* [NCCL_PROTO](env.html#nccl-proto)

  * [Values accepted](env.html#id68)

* [NCCL_NVB_DISABLE](env.html#nccl-nvb-disable)

  * [Value accepted](env.html#id69)

* [NCCL_PXN_DISABLE](env.html#nccl-pxn-disable)

  * [Value accepted](env.html#id70)

* [NCCL_P2P_PXN_LEVEL](env.html#nccl-p2p-pxn-level)

  * [Value accepted](env.html#id71)

* [NCCL_RUNTIME_CONNECT](env.html#nccl-runtime-connect)

  * [Value accepted](env.html#id72)

* [NCCL_GRAPH_REGISTER](env.html#nccl-graph-register)

  * [Value accepted](env.html#id74)

* [NCCL_LOCAL_REGISTER](env.html#nccl-local-register)

  * [Value accepted](env.html#id75)

* [NCCL_LEGACY_CUDA_REGISTER](env.html#nccl-legacy-cuda-register)

  * [Value accepted](env.html#id76)

* [NCCL_SET_STACK_SIZE](env.html#nccl-set-stack-size)

  * [Value accepted](env.html#id77)

* [NCCL_GRAPH_MIXING_SUPPORT](env.html#nccl-graph-mixing-support)

  * [Value accepted](env.html#id79)

* [NCCL_DMABUF_ENABLE](env.html#nccl-dmabuf-enable)

  * [Value accepted](env.html#id80)

* [NCCL_P2P_NET_CHUNKSIZE](env.html#nccl-p2p-net-chunksize)

  * [Values accepted](env.html#id81)

* [NCCL_P2P_LL_THRESHOLD](env.html#nccl-p2p-ll-threshold)

  * [Values accepted](env.html#id82)

__[NCCL](index.html)

* * *

NCCL and MPIÂ¶

APIÂ¶

The NCCL API and usage is similar to MPI but there are many minor differences. The following list summarizes these differences:

Using multiple devices per processÂ¶

Similarly to the concept of MPI endpoints, NCCL does not require ranks to be mapped 1:1 to processes. A NCCL communicator may have many ranks (and, thus, multiple devices) associated to a single process. Hence, if used with MPI, a single MPI rank (a NCCL process) may have multiple devices associated with it.

ReduceScatter operationÂ¶

The ncclReduceScatter operation is similar to the MPI_Reduce_scatter_block operation, not the MPI_Reduce_scatter operation. The MPI_Reduce_scatter function is intrinsically a â€œvectorâ€• function, while MPI_Reduce_scatter_block (defined later to fill the missing semantics) provides regular counts similarly to the mirror function MPI_Allgather. This is an oddity of MPI which has not been fixed for legitimate retro-compatibility reasons and that NCCL does not follow.

Send and Receive countsÂ¶

In many collective operations, MPI allows for different send and receive counts and types, as long as sendcount*sizeof(sendtype) == recvcount*sizeof(recvtype). NCCL does not allow that, defining a single count

and a single data-type.

For AllGather and ReduceScatter operations, the count is equal to the per-rank size, which is the smallest size; the other count being equal to nranks*count. The function prototype clearly shows which count is provided. ncclAllGather has a sendcount as argument, while ncclReduceScatter has a recvcount as argument.

Note: When performing or comparing AllReduce operations using a combination of ReduceScatter and AllGather, define the sendcount and recvcount as the total count divided by the number of ranks, with the correct count rounding-up, if it is not a perfect multiple of the number of ranks.

Other collectives and point-to-point operationsÂ¶

NCCL does not define specific verbs for sendrecv, gather, gatherv, scatter, scatterv, alltoall, alltoallv, alltoallw, nor neighbor collectives. All those operations can be simply expressed using a combination of ncclSend, ncclRecv, and ncclGroupStart/ncclGroupEnd, similarly to how they can be expressed with MPI_Isend, MPI_Irecv and MPI_Waitall.

ncclRecv does not support the equivalent of MPI_ANY_SOURCE; a specific source rank must always be provided. Similarly, the provided receive count must match the send count. Further, there is no concept of message tags.

In-place operationsÂ¶

For more information, see [In-place Operations](usage/inplace.html#in-place-operations).

Using NCCL within an MPI ProgramÂ¶

NCCL can be easily used in conjunction with MPI. NCCL collectives are similar to MPI collectives, therefore, creating a NCCL communicator out of an MPI communicator is straightforward. It is therefore easy to use MPI for CPU-to-CPU communication and NCCL for GPU-to-GPU communication.

However, some implementation details in MPI can lead to issues when using NCCL inside an MPI program.

MPI ProgressÂ¶

MPI defines a notion of progress which means that MPI operations need the program to call MPI functions (potentially multiple times) to make progress and eventually complete.

In some implementations, progress on one rank may need MPI to be called on another rank. While this is usually bad for performance, it can be argued that this is a valid MPI implementation.

As a result, blocking on a NCCL collective operation, for example calling cudaStreamSynchronize, may create a deadlock in some cases because not calling MPI on one rank could block other ranks, preventing them from reaching the NCCL call that would unblock the NCCL collective on the first rank.

In that case, the cudaStreamSynchronize call should be replaced by a loop like the following:

```
    cudaError_t err = cudaErrorNotReady;
    int flag;
    while (err == cudaErrorNotReady) {
      err = cudaStreamQuery(args->streams[i]);
                MPI_Iprobe(MPI_ANY_SOURCE, MPI_ANY_TAG, MPI_COMM_WORLD, &flag,
MPI_STATUS_IGNORE);
    }
```

Inter-GPU Communication with CUDA-aware MPIÂ¶

Using NCCL to perform inter-GPU communication concurrently with CUDA-aware MPI may create deadlocks.

NCCL creates inter-device dependencies, meaning that after it has been launched, a NCCL kernel will wait (and potentially block the CUDA device) until all ranks in the communicator launch their NCCL kernel. CUDA-aware MPI may also create such dependencies between devices depending on the MPI implementation.

Using both MPI and NCCL to perform transfers between the same sets of CUDA

devices concurrently is therefore not guaranteed to be safe.

[Next](env.html "Environment Variables") [Previous](examples.html "Examples")

* * *