    * [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)
    * [ncclGroupStart](../api/group.html#ncclgroupstart)
    * [ncclGroupEnd](../api/group.html#ncclgroupend)

* [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)

* * *

Group CallsÂ¶

Group functions (ncclGroupStart/ncclGroupEnd) can be used to merge multiple calls into one. This is needed for three purposes: managing multiple GPUs from one thread (to avoid deadlocks), aggregating communication operations to improve performance, or merging multiple send/receive point-to-point operations (see [Point-to-point communication](p2p.html#point-to-point) section). All three usages can be combined together, with one exception : calls to [`ncclCommInitRank()`](../api/comms.html#c.ncclCommInitRank "ncclCommInitRank") cannot be merged with others.

Management Of Multiple GPUs From One ThreadÂ¶

When a single thread is managing multiple devices, group semantics must be used. This is because every NCCL call may have to block, waiting for other threads/ranks to arrive, before effectively posting the NCCL operation on the given stream. Hence, a simple loop on multiple devices like shown below could block on the first call waiting for the other ones:

```
for (int i=0; i<nLocalDevs; i++) {
  ncclAllReduce(..., comm[i], stream[i]);
}
```

To define that these calls are part of the same collective operation, ncclGroupStart and ncclGroupEnd should be used:

```
ncclGroupStart();
for (int i=0; i<nLocalDevs; i++) {
  ncclAllReduce(..., comm[i], stream[i]);
}
ncclGroupEnd();
```

This will tell NCCL to treat all calls between ncclGroupStart and ncclGroupEnd as a single call to many devices.

Caution: When called inside a group, stream operations (like ncclAllReduce) can return without having enqueued the operation on the stream. Stream operations like cudaStreamSynchronize can therefore be called only after ncclGroupEnd returns.

Group calls must also be used to create a communicator when one thread manages more than one device:

```
ncclGroupStart();
for (int i=0; i<nLocalDevs; i++) {
  cudaSetDevice(device[i]);
  ncclCommInitRank(comms+i, nranks, commId, rank[i]);
```

```
}
ncclGroupEnd();
```

Note: Contrary to NCCL 1.x, there is no need to set the CUDA device before every NCCL communication call within a group, but it is still needed when calling ncclCommInitRank within a group.

Related links:

* [`ncclGroupStart()`](../api/group.html#c.ncclGroupStart "ncclGroupStart")
* [`ncclGroupEnd()`](../api/group.html#c.ncclGroupEnd "ncclGroupEnd")

Aggregated Operations (2.2 and later)Â¶

The group semantics can also be used to have multiple collective operations performed within a single NCCL launch. This is useful for reducing the launch overhead, in other words, latency, as it only occurs once for multiple operations. Init functions cannot be aggregated with other init functions, nor with communication functions.

Aggregation of collective operations can be done simply by having multiple calls to NCCL within a ncclGroupStart / ncclGroupEnd section.

In the following example, we launch one broadcast and two allReduce operations together as a single NCCL launch.

```
ncclGroupStart();
ncclBroadcast(sendbuff1, recvbuff1, count1, datatype, root, comm, stream);
ncclAllReduce(sendbuff2, recvbuff2, count2, datatype, comm, stream);
ncclAllReduce(sendbuff3, recvbuff3, count3, datatype, comm, stream);
ncclGroupEnd();
```

It is permitted to combine aggregation with multi-GPU launch and use different communicators in a group launch as shown in the Management Of Multiple GPUs From One Thread topic. When combining multi-GPU launch and aggregation, ncclGroupStart and ncclGroupEnd can be either used once or at each level. The following example groups the allReduce operations from different layers and on multiple CUDA devices :

```
ncclGroupStart();
for (int i=0; i<nlayers; i++) {
  ncclGroupStart();
  for (int g=0; g<ngpus; g++) {
      ncclAllReduce(sendbuffs[g]+offsets[i], recvbuffs[g]+offsets[i], counts[i], datatype[i], comms[g],
streams[g]);
  }
  ncclGroupEnd();
}
```

```
    ncclGroupEnd();
```

Note: The NCCL operation will only be started as a whole during the last call to ncclGroupEnd. The ncclGroupStart and ncclGroupEnd calls within the for loop are not necessary and do nothing.

Related links:

  * [`ncclGroupStart()`](../api/group.html#c.ncclGroupStart "ncclGroupStart")
  * [`ncclGroupEnd()`](../api/group.html#c.ncclGroupEnd "ncclGroupEnd")

Nonblocking Group OperationÂ¶

If a communicator is marked as nonblocking through ncclCommInitRankConfig, the group functions become asynchronous correspondingly. In this case, if users issue multiple NCCL operations in one group, returning from ncclGroupEnd() might not mean the NCCL communication kernels have been issued to CUDA streams. If ncclGroupEnd() returns ncclSuccess, it means NCCL kernels have been issued to streams; if it returns ncclInProgress, it means NCCL kernels are being issued to streams in the background. It is usersâ€™ responsibility to make sure the state of the communicator changes into ncclSuccess before calling related CUDA calls (e.g. cudaStreamSynchronize):

```
    ncclGroupStart();
```

```
    for (int g=0; g<ngpus; g++) {
        ncclAllReduce(sendbuffs[g]+offsets[i], recvbuffs[g]+offsets[i], counts[i], datatype[i], comms[g],
streams[g]);
    }
  ret = ncclGroupEnd();
  if (ret == ncclInProgress) {
    for (int g=0; g<ngpus; g++) {
      do {
        ncclCommGetAsyncError(comms[g], &state);
      } while (state == ncclInProgress);
    }
  } else if (ret == ncclSuccess) {
    /* Successfully issued */
    printf("NCCL kernel issue succeeded\n");
  } else {
    /* Errors happen */
    reportErrorAndRestart();
  }

  for (int g=0; g<ngpus; g++) {
    cudaStreamSynchronize(streams[g]);
  }
```

Related links:

* [`ncclCommInitRankConfig()`](../api/comms.html#c.ncclCommInitRankConfig

"ncclCommInitRankConfig")

* [`ncclCommGetAsyncError()`](../api/comms.html#c.ncclCommGetAsyncError "ncclCommGetAsyncError")

[Next](p2p.html "Point-to-point communication") [Previous](streams.html "CUDA Stream Semantics")

* * *