CnuasCCL, Datasheet

Collective communication library over the CnuasLink accelerator fabric

Document DS-CNU-017 • **Revision** C • **Issued** 15 August 2026 • **Status** Preview, v0.1 implemented over the bootstrap channel

Item	Value
Part	CnuasCCL
Type	Collective communication library
Models	NVIDIA NCCL
Status	v0.1 implemented, collectives carried over the bootstrap channel rather than the fabric
Repo	PacketFive/CnuasGPU

Partial implementation

v0.1 implements the bootstrap, the membership table, and broadcast, AllReduce and AllGather. It does **not yet move data over CnuasLink**: the collectives use the same caller supplied channel the bootstrap uses. The reason is in section 3. Multi accelerator training today still runs through stock NCCL over the RDMA network transport, exercised by `tests/test_nccl_link.py`.

1. Overview

CnuasCCL is the intended collective communication library for CnuasGPU, taking CnuasLink as its primary transport rather than the network. It is the accelerator side counterpart to NCCL and is what a distributed training run would call for gradient exchange.

2. Operations

Operation	Description	State
All reduce	Combine a buffer across all ranks and return the result to all	v0.1
Broadcast	Distribute one rank's buffer to all ranks	v0.1
All gather	Concatenate every rank's buffer at every rank	v0.1
Reduce scatter	Combine across ranks and distribute disjoint pieces	Specified
All to all	Exchange a distinct buffer between every pair of ranks	Specified

Element types are int8, uint8, int32, uint32, int64, uint64, float32 and float64; operators are sum, product, minimum and maximum.

AllReduce reduces in ascending rank order on every rank, starting from rank 0's contribution rather than from the local one. Floating point addition is not associative, so reducing in arrival order would give a different answer on each rank, and a result that depends on who asked is not an AllReduce. NCCL and MPI make the same guarantee, and `lib/test/cnuasccl_smoke` checks the outputs are bit identical rather than merely close.

2.1 Bootstrap

The communicator is built from a caller supplied blocking allgather, following UCC's `ucc_oob_coll` rather than NCCL's unique identifier plus rendezvous. This keeps the rendezvous outside the library, so a communicator

can be built with no control plane and no fabric, which is what makes the library testable at all. One allgather of a 16 byte record per rank yields the whole membership table.

The record is serialised field by field in little endian rather than copied as a struct, because two ranks can be separate processes built by different compilers.

Rank order is the caller's and need not follow `gpu_id` order, which is what leaves communicator splitting possible later. A ring is derived from the membership table rather than baked into it, so the tree and recursive doubling families stay available without a second allgather.

Every failure is reported through the return status. Nothing in the library aborts the process, which is a deliberate departure from most of the surveyed prior art: a library that kills its caller cannot have its failure paths tested. The decisions and the reasoning are recorded in section 7 of the bootstrap design.

3. Transport

Item	Specified value
Primary transport	CnuasLink, accelerator to accelerator
Fallback transport	RDMA network through CnuasNIC
Doorbell path	The fabric registers at BAR0 offset `0x200` on the accelerator, described in Kernel Modules

The hardware path already exists, and since milestone 8.2a a userspace wrapper for it ships in CnuasDev. v0.1 nonetheless carries collective payloads over the caller's bootstrap channel.

That is a staging decision rather than an oversight. Reading the driver and the switch shows the receive path has one staging buffer under one mutex, no per peer queues, no ordering and no delivery notification, so a frame from peer B can satisfy a thread waiting on peer A. A collective run directly over it would have to supply sequencing, retransmission and peer matching first. Building the API and pinning its semantics against a channel that already works means the transport can be replaced underneath without changing a caller. Section 3 of the bootstrap design lists the constraints with the evidence for each.

4. Dependencies

Requirement	Status
CnuasLink transport between accelerators	Implemented, see CnuasLink
Reduction arithmetic	Implemented for the shipped element wise kernels
Kernel launch for user written reductions	Not implemented, see CnuasCC
Ring and tree algorithm selection	Not yet selected; the membership table leaves both families open, options surveyed in the bootstrap design
Rank discovery and bootstrap	Implemented, milestone 8.2b, decisions in section 7 of the bootstrap design
In band collectives over CnuasLink	Not implemented; needs sequencing, retransmission and peer matching first
Userspace CnuasLink transport	Implemented, `cnuasdev_link_*` in the CnuasDev datasheet section 2a, roadmap milestone 8.2a

v0.1 is not blocked on the compiler, and is not built on it. The compiler is required only to widen the library to reductions the caller writes; the fixed operators above are computed on the host.

4.1 Build and test

`lib/libcnuasccl` builds `libcnuasccl.so` and depends on neither libcnuasrt nor a device, so it compiles and runs anywhere. `lib/test/cnuasccl_smoke` covers it with 71 checks and runs under `make -C lib check`.

The test runs its ranks as real threads through a barrier based allgather rather than looping over ranks in one thread. A single threaded stand in would let a rank observe the table before its peers had contributed, so it would pass whether or not the callback is genuinely a rendezvous.

5. Interim position

Until CnuasCCL moves data over the fabric, multi accelerator training runs with NCCL over the RDMA transport. That path is validated end to end in the superproject test suite, and it is how the multi accelerator behaviour of the platform is demonstrated today.

6. Revision history

Revision	Notes
A	First publication as a preview. Operation list taken from the CnuasGPU design document, section 7.6.
B	Records the userspace CnuasLink transport of milestone 8.2a as implemented, and places a first CnuasCCL at milestone 8.2c.
C	Records milestones 8.2b and 8.2c. The bootstrap, membership table, broadcast, AllReduce and AllGather are implemented. New section 2.1 on the bootstrap, and section 3 now explains why v0.1 does not yet use CnuasLink.

PacketFive, Packet Five Networks Ltd., Dublin, Ireland.