CnuasCCL, Datasheet

Collective communication library over the CnuasLink accelerator fabric

Document DS-CNU-017 • **Revision** D • **Issued** 23 August 2026 • **Status** Preview, v0.2 implemented over CnuasLink

Item	Value
Part	CnuasCCL
Type	Collective communication library
Models	NVIDIA NCCL
Status	v0.2 implemented, collective payload carried over CnuasLink, bootstrap retained for setup only
Repo	PacketFive/CnuasGPU

Fabric native payload since v0.2

v0.2 moves every collective payload byte over an injectable CnuasLink transport. The bootstrap channel is used for exactly two allgathers per communicator at setup (membership record and comm_id nonce) and is not reached by any subsequent collective. A communicator built without a transport returns `NOT_SUPPORTED` from every multi rank collective rather than silently reusing the bootstrap channel, so an accidentally bootstrap only build cannot pass the test suite.

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	v0.2
Broadcast	Distribute one rank's buffer to all ranks	v0.2
All gather	Concatenate every rank's buffer at every rank	v0.2
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_link_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. Two allgathers are run at init and no more: a 16 byte membership record per rank, then a 4 byte nonce per rank that is XORed into the per communicator `comm_id` used to stamp every payload frame.

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.

2.2 Payload envelope

Every payload frame carries a 40 byte envelope inside a `CNUASLINK_TYPE_DATA` frame. The envelope is byte wise little endian and holds `magic` (`"CCLE"`), `version`, `coll_kind`, `comm_id`, `seq`, `src_rank`, `dst_rank`, `chunk_index`, `chunk_count`, `chunk_bytes` and `total_bytes`. The receive path validates every one of those fields against what the collective expected, so a malformed, wrong communicator, wrong collective, out of sequence, wrong source, wrong chunk, duplicate or oversized frame is rejected with a dedicated status rather than silently accepted.

Chunk size is capped at 60 KiB so the envelope plus one chunk plus the CnuasLink frame header fits under the fabric maximum, and payloads larger than a chunk are split into a bounded number of chunks with the total size stamped on every frame.

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

v0.2 defines an injectable `cnuasccl_link_transport` (`send`, `recv`, `local_gpu` plus a caller owned `context`) that the library uses for every payload byte. The production transport wraps `cnuasdev_link_send` and `cnuasdev_link_recv` from the CnuasDev datasheet; the host test suite plugs in a fake switch instead, which is what makes the receive path testable without a device.

The receive path in the driver has one staging buffer under one mutex and no per peer queue. The library therefore demultiplexes frames by source at the collective layer: a frame from a peer that raced ahead is buffered in a per source FIFO until that peer's collective is due, rather than being confused for the frame the current recv was waiting for. The schedule is round robin unicast per peer so at most one in flight frame exists per receiver, matching the driver's single staging buffer. Section 3 of the bootstrap design records the underlying constraints and section 7 records how the envelope resolves each of them.

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

Requirement	Status
Ring and tree algorithm selection	Round robin unicast in v0.2; the membership table leaves ring, tree and recursive doubling open for later selection without a second allgather
Rank discovery and bootstrap	Implemented, milestone 8.2b, decisions in section 7 of the bootstrap design
In band collectives over CnuasLink	Implemented, v0.2, milestone 8.2d
Userspace CnuasLink transport	Implemented, `cnuasdev_link_*` in the CnuasDev datasheet section 2a, roadmap milestone 8.2a

v0.2 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 the bootstrap and public ABI; `lib/test/cnuasccl_link_smoke` covers the fabric native payload path with a fake CnuasLink transport that counts every byte on both channels, so a regression that reaches for bootstrap for payload cannot pass. The link smoke test pins the envelope wire layout byte by byte, checks broadcast, all reduce and all gather on multiple ranks, multiple rounds, multiple communicators, and chunking, and drives each envelope guard through its dedicated status. Both suites run 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

Multi accelerator training over CnuasCCL now moves its bytes over the fabric. Comparative benchmarks against NCCL over the RDMA transport remain in the superproject test suite because the reference NCCL path is what the platform is compared against.

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.
D	Records milestone 8.2d. v0.2 moves every collective payload byte over an injectable CnuasLink transport with a 40 byte envelope, per source demultiplexing, and bounded chunking. Bootstrap is used at init only. New section 2.2 pins the envelope, and section 3 records how the receive path is demultiplexed. Dependencies row for in band collectives now marked implemented.

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

Cnuas Virtual AI/HPC Infrastructure is emulation software. It is not affiliated with, endorsed by, or derived from any hardware vendor, and every device it models is a software artefact. The work is published by its authors in a personal capacity and is not sponsored or endorsed by any employer.

Specifications describe the referenced revision of the software and may change without notice. Contact info@packetfive.com.