

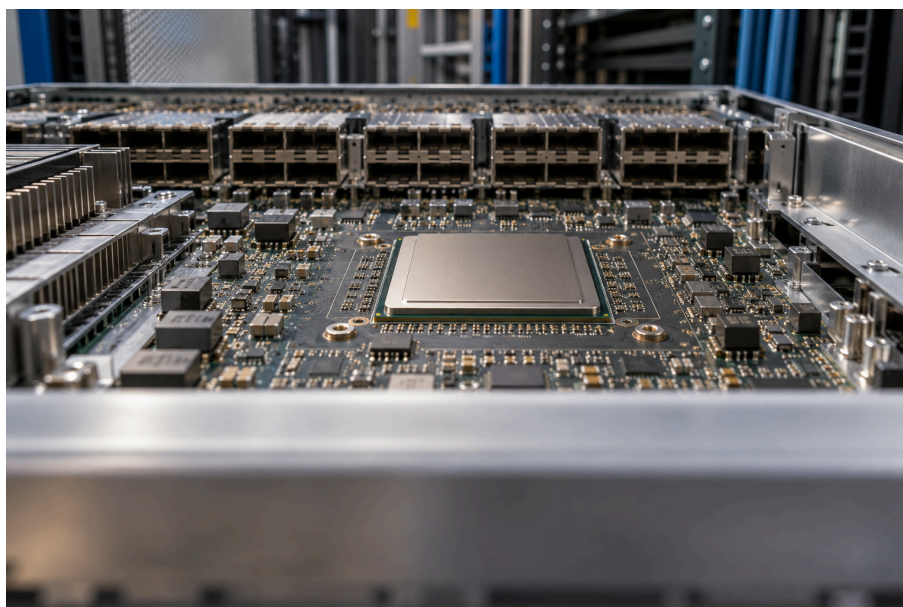


NVIDIA SHARP: In-Network All-Reduce, NVLS and CollNet

Plan and build private AI compute with GPU Smith. We connect workload requirements with GPU hardware, networking, deployment and operating decisions so your infrastructure fits the work it must perform.

Published by GPU Smith · 2026-10-29 · nccl nvls · nvidia sharp in-network computing · sharp allreduce infiniband · nvlink sharp nccl · nccl collnet enable · sharpv3 sharpv4 quantum-x800 · nccl sharp plugin requirements

Original article: <https://gpusmith.com/articles/en/nvidia-sharp-in-network-reduction-allreduce>



In this article

1. The all-reduce cost problem
2. What SHARP does
3. Two domains: NVLink SHARP and InfiniBand SHARP
4. SHARP generations
5. NCCL integration
6. What the published gains are
7. Precision caveats
8. Requirements and deployment checklist
9. When SHARP is not worth it
10. Validation and troubleshooting
11. Sources

The all-reduce cost problem

A data-parallel training step ends with an all-reduce: every rank holds a gradient buffer of size S and must end up with the element-wise sum of all n buffers. In NCCL's ring algorithm, GPU streaming multiprocessors (SMs) do that arithmetic, and the data crosses the fabric twice, once for reduce-scatter and once for all-gather. The `nccl-tests` performance notes put the ideal ring time at $t = (S/B) * (2*(n-1)/n)$ for link bandwidth B , and define bus bandwidth as $\text{busbw} = \text{algbw} * (2*(n-1)/n)$ so that results can be compared against hardware peak (`nccl-tests`: [PERFORMANCE.md](#)). The factor $2(n-1)/n$ is the problem: each rank sends, and receives, almost twice its buffer.

n (ranks in ring)	$2(n-1)/n$	Bytes sent per rank for $S = 1$ GB (illustrative)
2	1.000	1.000 GB
8	1.750	1.750 GB
64	1.969	1.969 GB
1,024	1.998	1.998 GB

Table 1. Ring all-reduce traffic per rank. Derived from the `nccl-tests` formula; $S = 1$ GB is an illustrative buffer size.

The ring also has $2(n-1)$ dependent steps, so per-hop latency dominates small messages and any slow rank stalls the rest, and its kernels occupy SMs that could run overlapping compute. NVIDIA's NCCL 2.27 post states that "traditional ring-based implementations can consume 16 or more SMs" ([NVIDIA: NCCL 2.27](#)). SHARP addresses all three costs (bytes, steps and SMs) by moving the summation out of the GPUs and into the switches.

What SHARP does

The Scalable Hierarchical Aggregation and Reduction Protocol "improves the performance of MPI and Machine Learning collective operation, by offloading collective operations from CPUs and GPUs to the network and eliminating the need to send data multiple times between endpoints" ([NVIDIA SHARP User Manual v3.0](#)). The logic is "incorporated in the switch ASIC" and performs reductions "directly within the network fabric". According to NVIDIA, moving the collective into the switch fabric "reduces the amount of transferred data by half and minimizes jitter" ([NVIDIA: Advancing Performance with SHARP](#)).

Each endpoint sends its contribution once up an aggregation tree of switches; each switch sums its children's streams and forwards one partial result, and the root's result is replicated back down to every endpoint. Data volume shrinks "as aggregation nodes are reached" ([SHARP User Manual v3.0](#)). Each endpoint therefore sends and receives about S instead of about $2S$, which is where the "half" comes from.

Two domains: NVLink SHARP and InfiniBand SHARP

SHARP exists in two places, which NCCL treats as different algorithms.

NVLink SHARP (NVLS) runs inside NVSwitch. Third-generation NVSwitch added "hardware acceleration for collective operations with multicast and NVIDIA SHARP in-network reductions". Embedded ALUs offer "up to 400 GFLOPS of FP32 throughput", and a SHARP controller "can manage up to 128 SHARP groups in parallel" ([NVIDIA: Third-Generation NVSwitch](#)). NCCL documents NVLS as available "in third-generation NVSwitch systems (NVLink4) with Hopper and later GPU architectures" ([NCCL environment variables](#)).

InfiniBand SHARP runs in Quantum-family InfiniBand switches. It reduces across nodes, and NCCL reaches it through the CollNet path and an external network plugin.

NCCL 2.18.1 combined the two domains. It added the inter-node algorithms "NVLink SHARP + IB SHARP (NVLS)" and "NVLink SHARP + Tree (NVLSTREE)" ([NCCL 2.18.1 release notes](#)). In the hierarchical case, NVSwitch reduces within the node, IB switches reduce across nodes, and the result is multicast back down both levels.

Worked example: NVLS inside one eight-GPU H100 node

NVIDIA's NVSwitch post counts the operations. A ring-style NCCL all-reduce costs " $2N$ reads and $2N$ writes at each GPU interface, or $4N$ total operations", while with SHARP it costs " $2N+2$ operations, or approximately halving the number of read/write operations" ([NVIDIA: Third-Generation NVSwitch](#)). In units of S/N per GPU, that is $(N+1)/N \times S$ in each direction for NVLS, compared with $2(N-1)/N \times S$ for the ring.

Assumptions (illustrative): $N = 8$ H100 GPUs, $S = 1$ GB, and 450 GB/s per direction per GPU. The 450 GB/s is derived as half of the 900 GB/s NVLink 4 bandwidth per GPU ([NVIDIA NVLink](#)) and matches NVIDIA's "450 GB/s of bandwidth for reduction operations" for an eight-H100 system ([NVIDIA: Third-Generation NVSwitch](#)).

1. Ring traffic per GPU = $2 \times 7/8 \times 1$ GB = 1.75 GB. Time = $1.75 / 450 = \mathbf{3.89\ ms}$.
2. NVLS traffic per GPU = $9/8 \times 1$ GB = 1.125 GB. Time = $1.125 / 450 = \mathbf{2.50\ ms}$.
3. Ratio = $3.89 / 2.50 \approx 1.56\times$ (derived, wire-time only).

4. nccl-tests would report $\text{algbw} = 1 \text{ GB} / 2.50 \text{ ms} = 400 \text{ GB/s}$ for NVLS. It would then report $\text{busbw} = 400 \times 1.75 = \mathbf{700 \text{ GB/s}}$.

Because nccl-tests applies the ring factor whatever algorithm ran, an NVLS or CollNet run can show busbw above the per-direction link rate; that indicates in-network reduction, not a measurement error. All four figures are upper bounds that ignore protocol overhead and synchronization.

Worked example: IB SHARP across 128 nodes

Assumptions (illustrative): 128 nodes $\times$ 8 GPUs = 1,024 ranks, one 400 Gb/s NIC per GPU (8 rails per node), treated as 50 GB/s per NIC with encoding and protocol overhead ignored, and $S = 1 \text{ GB}$.

1. **Ring.** Every ring channel enters and leaves each node once, and each link in a channel carries $2(n-1)/n$ of that channel's slice. Summed over channels, a node's network egress is $2 \times 1023/1024 \times 1 \text{ GB} \approx 1.998 \text{ GB}$, or $\approx 0.250 \text{ GB}$ per NIC. Time $\approx 0.250 / 50 = \mathbf{5.0 \text{ ms}}$.
2. **Hierarchical IB SHARP.** An intra-node reduce-scatter over NVLink leaves each GPU holding a reduced $1/8$ slice. Each NIC sends 0.125 GB up its rail's aggregation tree and receives 0.125 GB back. Time $\approx 0.125 / 50 = \mathbf{2.5 \text{ ms}}$, plus the intra-node phases on the faster NVLink domain.
3. The inter-node byte reduction is $\approx 2\times$, which matches NVIDIA's "by half" statement. The step count also changes: from $2(n-1) = 2,046$ dependent ring steps to a round trip through a few switch tiers.

NCCL normally picks its own Tree algorithm for small messages, so SHARP's real latency advantage is smaller than the ring comparison suggests; see the guide to [NCCL algorithms and protocols](#).

SHARP generations

Generation	Introduced with	What NVIDIA says it added
SHARPV1	EDR 100 Gb/s switches	Designed for scientific computing; MVAPICH2 on TACC Frontera reported 5x MPI AllReduce and up to 9x MPI Barrier
SHARPV2	HDR 200 Gb/s Quantum	Large-message reduction, "a single workload at a time"
SHARPV3	Quantum-2 NDR 400G	Multi-tenant in-network computing; "multiple aggregation trees can be built over the same topology"
SHARPV4	Quantum-X800 XDR	"New algorithms to support a larger variety of collective communications"; FP8 and new collectives per the datasheet

Table 2. SHARP generations. Sources: [NVIDIA SHARP blog](#), [SHARP User Manual v3.0](#), [Quantum-X800 datasheet](#).

SHARPV4 needs a status check because NVIDIA's wording changed. The October 2024 SHARP blog says SHARPV4 "will be released with the NVIDIA Quantum-X800 XDR InfiniBand switch platforms" ([NVIDIA SHARP blog](#)). The March 2024 launch release claimed "a 9x increase of 14.4Tflops of In-Network Computing" with SHARPV4 and said the platform would be available "next year" ([NVIDIA newsroom, 18 March 2024](#)). Later documents use the present tense:

- The NVIDIA datasheet (AUG25, hosted by Dell) says the fourth generation "adds support for FP8 precision and new collective operations such as ReduceScatter and ScatterGather" ([Quantum-X800 datasheet](#)).
- The product page says the switch "includes hardware-based In-Network Computing with SHARP v4" ([NVIDIA Quantum-X800](#)).
- NVIDIA's validated-configuration matrix for XDR clusters lists Q3400-RA and Q3200-RA switches alongside SHARP software versions 3.10.3 through 3.16.3 ([NVIDIA Quantum-X800 \(XDR\) Clusters](#)).
- The SHARP software's supported-device list includes Quantum-3, with the footnote "Support limited to NVIDIA Quantum-3 based XDR Q34xx-RA InfiniBand Switch" ([SHARP: General Information](#)).

As of 2026-10-28, Quantum-X800 is documented as a shipping, validated platform with SHARP software support. This guide found no NVIDIA document stating which SHARPV4 features (FP8 in-switch reduction, the new collectives) a given SHARP and NCCL release exposes. The newest public SHARP change log, 3.15 (June 2026), adds a beta parallel All-Gather/Reduce-Scatter over InfiniBand multicast and does not mention FP8 ([SHARP 3.15: Changes and New Features](#)). Confirm against the release notes of the exact build.

NCCL integration

NCCL reaches the two domains through different controls. The defaults below come from the NCCL 2.32.3 environment-variable reference (NCCL v2.32.3-1, released 2026-09-17 and still the latest release on 2026-10-28; [NCCL environment variables](#)).

Variable	Since	Default	Meaning
NCCL_NVLS_ENABLE	2.17	2	0 disables NVLS. 1 and 2 both proceed if NVLS is unsupported but fail if NVLS resources cannot be allocated
NCCL_ALGO values	2.14 / 2.17 / 2.18	auto	CollnetChain, CollnetDirect (2.14+), NVLS (2.17+), NVLSTree (2.18+)
NCCL_COLLNET_ENABLE	2.6	0	"define and set to 1 to use the CollNet plugin"
NCCL_COLLNET_NODE_THRESHOLD	2.9.9	2	Node count below which CollNet is not enabled
NCCL_IGNORE_COLLNET_MISMATCH	n/a	0	Fail initialization if ranks see different CollNet device counts
NCCL_DEBUG_SUBSYS	n/a	INIT,BOOTSTRAP,ENV	Add NVLS, TUNING, REG to trace selection and registration

Table 3. NCCL controls relevant to SHARP. Source: [NCCL environment variables](#).

Three details about NCCL_NVLS_ENABLE matter in practice. In 2.27 and 2.28, the default value 2 tried to fall back silently when NVLS allocation failed. That "could result in a hang if the allocation failed on just some of the ranks", so 2.29 reverted to failing, as value 1 does. The same reference states that NVLS is not compatible with multiple ranks of one communicator sharing a GPU. In that case, value 1 fails initialization and the default 2 silently disables NVLS ([NCCL environment variables](#)).

IB SHARP requires the NCCL-SHARP plugin. NVIDIA's SHARP manual says the plugin ships with HPC-X and loads automatically in HPC-X module environments. It can also be built from the `nccl-rdma-sharp-plugins` repository ([SHARP: Using SHARP with NCCL](#); [nccl-rdma-sharp-plugins](#)). Streaming aggregation is supported on one communicator at a time, so `NCCL_SHARP_DISABLE` (per process group) and `NCCL_SHARP_GROUP_SIZE_THRESH` choose where it is used, and `NCCL_IBEXT_DISABLE` falls back to native transports. `SHARP_COLL_LOCK_ON_COMM_INIT=1` and `SHARP_COLL_NUM_COLL_GROUP_RESOURCE_ALLOC_THRESHOLD=0` reserve resources at initialization and are on by default from plugin 2.1.x.

Buffer registration determines how much SM work disappears. NVLS user-buffer registration has been available since 2.19. It requires `ncclMemAlloc()` or a qualified VMM allocator and identical buffer offsets across ranks. IB SHARP registration has been available since 2.21 and "can reduce the number of NCCL SM usage down to 1". It is currently limited to communicators with one rank per node, and network registration requires `NCCL_PXN_DISABLE=1` because PXN is on by default ([NCCL buffer registration](#)). NCCL 2.27 extended SHARP to AllGather and ReduceScatter "for both NVLink and IB fabrics". It noted that these collectives "are now preferred over AllReduce" for large LLM training ([NVIDIA: NCCL 2.27](#)).

What the published gains are

Claim	Context	Source
5x MPI AllReduce, up to 9x MPI Barrier	SHARPV1, MVAPICH2, TACC Frontera; MPI, not NCCL	SHARP blog
2x AllReduce bandwidth → 17% higher BERT training	SHARPV2	SHARP blog
"Nearly an order of magnitude" AllReduce latency benefit	SHARPV3 on Azure	SHARP blog
Up to 2x throughput for small-block collectives	NVSwitch reductions vs NCCL on A100; marked preliminary	Hopper in-depth
16+ SMs reduced to 6 or fewer (2.7x)	NCCL 2.27 AllGather/ReduceScatter with NVLink + IB SHARP	NCCL 2.27
SM usage down to 1	IB SHARP buffer registration, one rank per node	NCCL buffer registration

Table 4. Vendor-published SHARP results, with their context.

These are vendor numbers on vendor-chosen configurations, and two figures often quoted alongside them are not SHARP results. The NCCL 2.27 "up to 9x reduction in latency for small message sizes" and "up to 2.5x higher performance" on NVL8 come from the symmetric-memory low-latency kernels, not from in-network reduction ([NVIDIA: NCCL 2.27](#)). Application-level gains such as the 17% BERT figure depend on how much of the step is exposed communication.

Precision caveats

In-network reduction moves the arithmetic to an ALU the framework does not control. For the NCCL 2.27 symmetric kernels, NVIDIA states that "reductions are computed using FP32 accumulators (or FP16 for FP8 on NVLink Switch (NVLS) Systems)" ([NVIDIA: NCCL 2.27](#)). An FP8 all-reduce on NVLS therefore accumulates at lower precision. On the IB side, SHARP 3.14 added a `sharp_am` parameter, `rounding_method`, defaulting to 0 (round to nearest even), with 4 selecting LFSR stochastic rounding, for all SHARP jobs ([SHARP: Changes and New Features](#)). That setting is fabric-wide and owned by the fabric administrator, not the job. The NCCL 2.18.1 notes list a hang for bfloat16 reductions with IB SHARP "on systems with less than 1 NIC per GPU" ([NCCL 2.18.1 release notes](#)). A prudent operator compares loss curves, or bitwise outputs where determinism matters, with SHARP on and off before relying on it for low-precision collectives.

Requirements and deployment checklist

Item	Requirement
IB switches	Quantum, Quantum-2, Quantum-3 (XDR Q34xx-RA only)
HCAs	ConnectX-6 and above; the NCCL page asks for ConnectX-6 HDR and a Quantum HDR switch or later
Host stack	DOCA-Host, GPUDirect RDMA (<code>nv_peer_mem</code>), NCCL $\geq 2.7.3$, plugin via HPC-X
Packaging	SHARP "is no longer provided as part of the DOCA-Host package" (3.14)
Fabric software	Aggregation Manager (<code>sharp_am</code>); UFM/OpenSM versions listed per SHARP release
Topology	One aggregation flow per HCA rail; group leaf switches by rail
Multi-tenancy	SHARPV3+ for concurrent jobs; v2 serves one workload at a time
NVLS	NVSwitch gen 3 + Hopper or later; one rank per GPU; VMM buffers for registration

Table 5. Deployment checklist. Sources: [SHARP General Information](#), [Using SHARP with NCCL](#), [Changes and New Features](#), [SHARP blog](#), [NCCL env](#), [Buffer registration](#).

Rail grouping interacts with fabric design, so plan the SHARP tree layout together with the [rail and oversubscription sizing](#) rather than added afterwards.

When SHARP is not worth it

IB SHARP is an InfiniBand feature. The SHARP supported-device list names only Quantum-family switches ([SHARP General Information](#)), and NVIDIA markets Quantum InfiniBand as "the world's only fully offloadable, In-Network Computing platform" ([NVIDIA InfiniBand](#)). A RoCE or other Ethernet scale-out fabric therefore runs GPU-based algorithms between nodes, though NVLS can still help inside the NVLink domain. Whether that difference justifies InfiniBand is one input among many in the [InfiniBand vs Ethernet for AI](#) decision.

SHARP also adds little in several other situations:

- **Single-node or two-node jobs.** CollNet is not enabled below `NCCL_COLLNET_NODE_THRESHOLD`, which defaults to 2.
- **Compute-bound steps.** Where communication is already hidden behind backward compute, little is left to save.
- **Multiple ranks per GPU.** NVLS is disabled in this configuration.
- **Multi-tenant clusters on SHARPV2 hardware.** Only one job at a time gets aggregation.
- **PXN dependence.** Software stacks that rely on PXN cannot also use network buffer registration.

There, better compute and communication overlap and NCCL's own hierarchical algorithms are the levers.

Validation and troubleshooting

Run A/B comparisons on the same nodes and the same NCCL build:

1. Run `all_reduce_perf` from `nccl-tests` with `NCCL_COLLNET_ENABLE=0` and then `=1`, and with `NCCL_NVLS_ENABLE=0` and then the default. Sweep message sizes from kilobytes to gigabytes.
2. Set `NCCL_DEBUG=INFO` and `NCCL_DEBUG_SUBSYS=INIT,NVLS,TUNING,REG` to confirm which algorithm ran and whether registration succeeded. Add `SHARP_COLL_LOG_LEVEL=3` for the plugin.
3. Read busbw with the arithmetic above in mind. A busbw above the per-direction link rate suggests an in-network path. A busbw identical to the baseline suggests that SHARP was never selected.
4. Measure step time and exposed communication in the real training job. Freed SMs only pay off if the framework uses them.

An NVLS initialization failure usually points to resource allocation, and partial-rank failures on 2.27 and 2.28 could hang instead of erroring; those cases call for the [NCCL hang triage](#) procedure rather than further tuning.

GPU Smith can review SHARP and NVLS enablement as part of a broader fabric assessment.

Sources

1. [Advancing Performance with NVIDIA SHARP In-Network Computing](#) - NVIDIA Technical Blog (Oct 2024), generations v1-v4, halving data and jitter, 17% BERT, Azure latency, MVAICH2 results (accessed 2026-10-28)
2. [Enabling Fast Inference and Resilient Training with NCCL 2.27](#) - NVIDIA Technical Blog (Jul 2025), SHARP for AllGather/ReduceScatter, SM figures, FP32/FP16 accumulators, symmetric-memory 9x/2.5x context (accessed 2026-10-28)
3. [NVIDIA SHARP User Manual v3.0](#) - NVIDIA, SHARP definition, aggregation nodes, multiple aggregation trees in v3 (accessed 2026-10-28)
4. [Using NVIDIA SHARP with NVIDIA NCCL](#) - NVIDIA SHARP docs, plugin requirements, NCCL and SHARP variables, rail grouping (accessed 2026-10-28)
5. [SHARP: General Information](#) - NVIDIA SHARP docs, supported switches, HCAs, packages, Q34xx-RA footnote (accessed 2026-10-28)

6. [SHARP: Changes and New Features](#) - NVIDIA SHARP docs 3.14, rounding_method, stochastic rounding, DOCA-Host packaging change (accessed 2026-10-28); 3.15 edition (last updated 2026-06-03), parallel AG/RS beta: <https://networking-docs.nvidia.com/sharpum/3150/changes-and-new-features>
7. [NCCL Environment Variables](#) - NVIDIA NCCL docs, NVLS/CollNet/ALGO/DEBUG_SUBSYS defaults and semantics (accessed 2026-10-28)
8. [NCCL User Buffer Registration](#) - NVIDIA NCCL docs, NVLS and IB SHARP registration, PXN incompatibility, SM usage down to 1 (accessed 2026-10-28)
9. [NCCL Release 2.18.1](#) - NVIDIA, NVLS/NVLSTree inter-node algorithms, bfloat16 IB SHARP known issue (accessed 2026-10-28)
10. [nccl-tests PERFORMANCE.md](#) - NVIDIA on GitHub, algbw/busbw definitions and 2(n-1)/n factor (accessed 2026-10-28)
11. [Upgrading Multi-GPU Interconnectivity with the Third-Generation NVIDIA NVSwitch](#) - NVIDIA Technical Blog (Aug 2022), NVSwitch SHARP ALUs, 128 groups, 2N+2 vs 4N operations, 450 GB/s reduction bandwidth (accessed 2026-10-28)
12. [NVIDIA Hopper Architecture In-Depth](#) - NVIDIA Technical Blog (Mar 2022), NVSwitch in-network reductions, up to 2x small-block claim (accessed 2026-10-28)
13. [NVIDIA NVLink and NVLink Switch](#) - NVIDIA, NVLink bandwidth per GPU by generation (accessed 2026-10-28)
14. [NVIDIA Announces New Switches Optimized for Trillion-Parameter GPU Computing and AI Infrastructure](#) - NVIDIA Newsroom (Mar 2024), SHARPV4 14.4 TFLOPS / 9x claim, availability "next year" (accessed 2026-10-28)
15. [NVIDIA Quantum-X800 InfiniBand Platform](#) - NVIDIA, SHARP v4 in-network computing, 144 x 800 Gb/s ports (accessed 2026-10-28)
16. [NVIDIA Quantum-X800 InfiniBand Switches Datasheet](#) - NVIDIA datasheet (AUG25) hosted by Dell, 4th-gen SHARP, FP8 and new collectives (accessed 2026-10-28)
17. [NVIDIA Quantum-X800 \(XDR\) Clusters](#) - NVIDIA validated configuration, Q3400-RA/Q3200-RA with SHARP 3.10.3-3.16.3 (accessed 2026-10-28)
18. [NVIDIA InfiniBand Networking](#) - NVIDIA, "fully offloadable, In-Network Computing platform" (accessed 2026-10-28)
19. [nccl-rdma-sharp-plugins](#) - NVIDIA/Mellanox on GitHub, NCCL SHARP plugin source and build requirements (accessed 2026-10-28)

Source links

Links cited in this article, in order of first appearance. Full addresses are provided for readers using extracted text.

1. <https://github.com/NVIDIA/nccl-tests/blob/master/doc/PERFORMANCE.md>
2. <https://developer.nvidia.com/blog/enabling-fast-inference-and-resilient-training-with-nccl-2-27>
3. <https://networking-docs.nvidia.com/sharpum/300>
4. <https://developer.nvidia.com/blog/advancing-performance-with-nvidia-sharp-in-network-computing>
5. <https://developer.nvidia.com/blog?p=53977>
6. <https://docs.nvidia.com/deeplearning/nccl/user-guide/docs/env.html>

7. https://docs.nvidia.com/deeplearning/nccl/release-notes/rel_2-18-1.html
8. <https://www.nvidia.com/en-us/data-center/nvlink>
9. <https://www.delltechnologies.com/asset/en-my/products/networking/technical-support/nvidia-quantum-x800-q3200-ra-and-q3400-ra-datasheet.pdf>
10. <https://nvidianews.nvidia.com/news/networking-switches-gpu-computing-ai>
11. <https://www.nvidia.com/en-us/networking/products/infiniband/quantum-x800>
12. <https://networking-docs.nvidia.com/nvidia-quantum-x800-xdr-clusters>
13. <https://networking-docs.nvidia.com/sharpum/3140/general-information>
14. <https://networking-docs.nvidia.com/sharpum/3150/changes-and-new-features>
15. <https://networking-docs.nvidia.com/sharpum/3140/using-nvidia-sharp-with-nvidia-nccl>
16. <https://github.com/Mellanox/nccl-rdma-sharp-plugins>
17. <https://docs.nvidia.com/deeplearning/nccl/user-guide/docs/usage/bufferreg.html>
18. <https://developer.nvidia.com/blog/nvidia-hopper-architecture-in-depth>
19. <https://networking-docs.nvidia.com/sharpum/3140/changes-and-new-features>
20. <https://www.nvidia.com/en-us/networking/products/infiniband>

THE PUBLISHER

About GPU Smith

GPU Smith provides independent engineering and research for private AI infrastructure. We help technical buyers and operators reason about GPU workload sizing, hardware selection, procurement, deployment and inference operations, from the compute system to the networking, power and cooling requirements around it.

Start with the workload

Useful infrastructure decisions begin with the models, data, throughput, latency and operating constraints a team actually has. GPU Smith helps connect those requirements with system design and deployment choices. The goal is a privately controlled AI environment whose capacity and operational demands are understood before a purchasing decision.

Hardware and supplier research

Our public reference library covers GPUs, complete systems and networking components. The server-vendor directory and manufacturing research help teams investigate suppliers, compare options and follow sources behind technical claims. We distinguish manufacturer specifications from measured benchmarks and advertised capabilities from tested configurations. Reference pages are research resources, not a live inventory listing or a binding equipment quote.

Deployment and operations

GPU Smith's engineering scope includes infrastructure integration, networking, power, cooling and the practical operation of inference workloads. Our published articles explain the tradeoffs behind deployment, procurement and ongoing operations so teams can ask better questions and document decisions.

Work with GPU Smith

Explore the [hardware library](#), [GPU references](#), [networking references](#) and [vendor research](#). [Contact GPU Smith](#) with your workloads and deployment constraints to discuss a project and confirm current scope, pricing and availability.

Inclusion of a manufacturer or product in our research does not imply a partnership, certification or endorsement.

Website: <https://gpusmith.com>

This article is educational. Verify consequential medical, legal, financial, regulatory and technical decisions with appropriate professionals and the cited primary sources. Company services are described separately from the article's research findings.