

NVIDIA Collective Communication Library (NCCL)

Release Notes

RN-08645-000_v2.29.7 | November 2026

Table of Contents

Chapter 1. NCCL Overview 1
Chapter 2. NCCL Release 2.32.6 3
Chapter 3. NCCL Release 2.32.3 8
Chapter 4. NCCL Release 2.31.2 11
Chapter 5. NCCL Release 2.30.7 15
Chapter 6. NCCL Release 2.30.4 18
Chapter 7. NCCL Release 2.30.3 20
Chapter 8. NCCL Release 2.29.7 23
Chapter 9. NCCL Release 2.29.3 26
Chapter 10. NCCL Release 2.29.2 27
Chapter 11. NCCL Release 2.28.9 30
Chapter 12. NCCL Release 2.28.7 31
Chapter 13. NCCL Release 2.28.3 33
Chapter 14. NCCL Release 2.27.7 36
Chapter 15. NCCL Release 2.27.6 38
Chapter 16. NCCL Release 2.27.5 40
Chapter 17. NCCL Release 2.27.3 42
Chapter 18. NCCL Release 2.26.5 45
Chapter 19. NCCL Release 2.26.2 47
Chapter 20. NCCL Release 2.25.1 50
Chapter 21. NCCL Release 2.24.3 51
Chapter 22. NCCL Release 2.23.4 53
Chapter 23. NCCL Release 2.22.3 55
Chapter 24. NCCL Release 2.21.5 57
Chapter 25. NCCL Release 2.20.5 59
Chapter 26. NCCL Release 2.20.3 60
Chapter 27. NCCL Release 2.19.3 62
Chapter 28. NCCL Release 2.18.5 64
Chapter 29. NCCL Release 2.18.3 66

Chapter 30. NCCL Release 2.18.1 68
Chapter 31. NCCL Release 2.17.1 70
Chapter 32. NCCL Release 2.16.5 72
Chapter 33. NCCL Release 2.16.2 74
Chapter 34. NCCL Release 2.15.5 76
Chapter 35. NCCL Release 2.15.1 77
Chapter 36. NCCL Release 2.14.3 78
Chapter 37. NCCL Release 2.13.4 80
Chapter 38. NCCL Release 2.12.12 82
Chapter 39. NCCL Release 2.12.10 84
Chapter 40. NCCL Release 2.12.7 85
Chapter 41. NCCL Release 2.11.4 86
Chapter 42. NCCL Release 2.10.3 87
Chapter 43. NCCL Release 2.9.9 88
Chapter 44. NCCL Release 2.9.8 89
Chapter 45. NCCL Release 2.9.6 90
Chapter 46. NCCL Release 2.8.4 91
Chapter 47. NCCL Release 2.8.3 92
Chapter 48. NCCL Release 2.7.8 94
Chapter 49. NCCL Release 2.7.6 96
Chapter 50. NCCL Release 2.7.5 97
Chapter 51. NCCL Release 2.7.3 99
Chapter 52. NCCL Release 2.6.4 101
Chapter 53. NCCL Release 2.5.6 102
Chapter 54. NCCL Release 2.4.8 103
Chapter 55. NCCL Release 2.4.7 105
Chapter 56. NCCL Release 2.4.2 107
Chapter 57. NCCL Release 2.3.7 108
Chapter 58. NCCL Release 2.3.5 109
Chapter 59. NCCL Release 2.3.4 110
Chapter 60. NCCL Release 2.2.13 111

Chapter 61. NCCL Release 2.2.12 112
Chapter 62. NCCL Release 2.1.15 114
Chapter 63. NCCL Release 2.1.4 116
Chapter 64. NCCL Release 2.1.2 117
Chapter 65. NCCL Release 2.0.5 118
Chapter 66. NCCL Release 2.0.4 119
Chapter 67. NCCL Release 2.0.2 120

Chapter 1. NCCL Overview

The NVIDIA® Collective Communications Library ™ (NCCL) (pronounced “Nickel”) is a library of multi-GPU collective communication primitives that are topology-aware and can be easily integrated into applications.

Collective communication algorithms employ many processors working in concert to aggregate data. NCCL is not a full-blown parallel programming framework; rather, it is a library focused on accelerating collective communication primitives. The following collective operations are currently supported:

- `AllReduce`
- `Broadcast`
- `Reduce`
- `AllGather`
- `ReduceScatter`

Tight synchronization between communicating processors is a key aspect of collective communication. CUDA® based collectives would traditionally be realized through a combination of CUDA memory copy operations and CUDA kernels for local reductions. NCCL, on the other hand, implements each collective in a single kernel handling both communication and computation operations. This allows for fast synchronization and minimizes the resources needed to reach peak bandwidth.

NCCL conveniently removes the need for developers to optimize their applications for specific machines. NCCL provides fast collectives over multiple GPUs both within and across nodes. It supports a variety of interconnect technologies including PCIe, NVLink™ , InfiniBand Verbs, and IP sockets. NCCL also automatically patterns its communication strategy to match the system’s underlying GPU interconnect topology.

Next to performance, ease of programming was the primary consideration in the design of NCCL. NCCL uses a simple C API, which can be easily accessed from a variety of programming languages. NCCL closely follows the popular collectives API defined by MPI (Message Passing Interface). Anyone familiar with MPI will thus find NCCL API very natural to use. In a minor departure from MPI, NCCL collectives take a “stream” argument which provides direct integration with the CUDA programming model. Finally, NCCL is compatible with virtually any multi-GPU parallelization model, for example:

- single-threaded
- multi-threaded, for example, using one thread per GPU

- multi-process, for example, MPI combined with multi-threaded operation on GPUs

NCCL has found great application in deep learning frameworks, where the `AllReduce` collective is heavily used for neural network training. Efficient scaling of neural network training is possible with the multi-GPU and multi node communication provided by NCCL.

Chapter 2. NCCL Release 2.32.6

This is the NCCL 2.32.6 release notes. For previous NCCL release notes, refer to the NCCL Archives.

Compatibility

NCCL 2.32.6 has been tested with the following:

- Deep learning framework containers. Refer to the Support Matrix for the supported container version.
- This NCCL release supports CUDA 12.x and CUDA 13.x.

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- Vera Rubin Performance: Added a pipeline-based cost model with Vera Rubin tuning data for Ring, Tree, NVLS, and NVLSTree. Added Vera Rubin models for symmetric AllGather, ReduceScatter, and AllReduce kernels. Sped up the topology graph search on large NVLink domains such as 32 to 40 GPU VR200 systems, reducing communicator initialization time. Tuner plugin interface v7 adds Rubin platform support.
- Device API Enhancements: Added an experimental Flow device API for building pipelined collective kernels over LSA and GIN connections with the Simple, LL, and LL128 protocols; the Flow API signatures are not stable and may change. Extended `ncclGetPeerPointer` and the resource-buffer peer pointer to address every load/store-reachable local peer by world rank, and added `ncclDevCommCanGetPeerPointer` to report whether a peer is reachable. Added the `NCCL_GIN_CONNECTION_CUSTOM_ARRAY` GIN connection type so applications can request connectivity to an explicit list of peers. Added `ncclLsaCopyTma`, a TMA-based variant of the `ncclLsaCopy` broadcast on Blackwell and later GPUs. Added a signal-efficient two-signal GIN barrier that GIN plugins can request through the v15 barrierOptions property. CFT teams now require and use CUDA logical-endpoint clique information to determine unicast and multicast team sizes. Added validation of GPU and NIC capabilities when establishing GDAKI connections, falling back to the proxy backend when GDAKI is unsupported and `NCCL_GIN_TYPE` is unset.

Updated the GPI device backend for the latest `libgpi` device ABI, including 31-bit signal and counter IDs and extended transfer-size encoding, and updated the bundled GPUNetIO code to v5.0.0-rc3. Moved the IR-only C device ABI (`ncclGin_C`, `ncclCoopAny`) out of the public device headers into `bindings/ir`.

- Collectives and Runtime Enhancements: Added experimental collective kernels called General Kernels (GENK), built on top of the new Flow APIs. These kernels are compiled out by default to minimize binary size; build with `SYM_GENK=1` and set `NCCL_SYM_GENK_ENABLE=1` to try them. Added experimental `AlltoAll_LL`, a latency-oriented symmetric `AlltoAll` kernel for small and medium per-peer messages on LSA and mixed LSA/GIN communicators, selectable with `NCCL_SYM_KERNEL=AlltoAll_LL`. Added host RMA support on rail-only configurations such as `NCCL_CROSS_NIC=0`, where host RMA was previously disabled. Updated the GIN and RMA plugin interfaces to v15 and v17: connection setup receives an abort flag so `ncclCommAbort` can interrupt blocking GIN and RMA initialization, and the plugin `rankStride` is replaced by an explicit `peerArray`. Improved hierarchical CE collective bandwidth when inter-node traffic is distributed across multiple RMA contexts, including under CUDA graph capture. Reduced host munmap latency on ATS platforms such as GB200 by granting `ncclMemAlloc` memory access to the owning GPU only.
- Diagnostics and Profiling: Added profiler plugin interface v8 with opt-in NCCL-sourced GPU and CPU timestamps on every event. Added support for loading up to two profiler plugins at a time through a comma-separated `NCCL_PROFILER_PLUGIN`, each with its own event mask. Allowed NetPlugin profiler events to be collected independently of proxy events; profiler plugins must accept ProxyStep or NULL parents. Updated Inspector to record per-call profiler tags (requiring profiler v7 or later), emit the selected algorithm and protocol per collective in its JSON output, and ship a communication-pattern extractor; `NCCL_INSPECTOR_NCCL_SOURCED_TIMINGS` defaults to on. Added a RAS diagnostics check verifying that the network devices NCCL selected are consistent across the nodes of a communicator and within each node. Added the DIAG debug subsystem (`NCCL_DEBUG_SUBSYS=DIAG`) for active and RAS diagnostics details. Attached all GIN/GDAKI queue pairs to a dedicated hardware Q counter set so aggregate counters can be queried for GIN traffic; controlled by `NCCL_GIN_GDAKI_Q_COUNTER` (default on).
- Other Improvements: Enabled building NCCL for Windows on Arm64. Reduced library size by compressing device code with `--compress-mode=size` when building with CUDA 13 or newer. Added channel-major and step-major NVLS staging FIFO layouts that reduce NVLink uTLB pressure for large-message NVLS and NVLSTree on Rubin; `NCCL_NVLS_FIFO_LAYOUT` controls the selection. Built `libnccl_device.bc` and `libnccl_device.ltoir` with Clang/LLVM 23, and unified the IR build options as `IR_ARCH` and `NCCL_EMIT_LLVM_IR`. Relaxed `ncclCommGrow` so existing ranks may pass either the original `ncclUniqueId` or NULL. Added a thread-exclusive Put path to the EFA-GDA GIN backend and printed endpoint counts when a flush times out (Github PR #2478), (Github PR #2552). Invoked the build-time Python generators through

python3, and resolved Git version metadata from the NCCL source tree for out-of-tree CMake builds (Github PR #2452), (Github PR #2451). Added a user-guide page on the determinism of NCCL reductions, and improved various other documentation.

- Contrib/Updates: Moved `nccl_ep` and `nccl_m2n` out of the NCCL repository; they are now developed at https://github.com/NVIDIA/nccl-extensions. Added `contrib/tide`, a community contribution for layout-aware FSDP2 collectives, built on top of NCCL device and host APIs. Updated `nccl4rust` for the NCCL 2.32 API and added a minimal AllReduce example.

Fixed Issues

The following issues have been resolved in NCCL 2.32.6:

- Fixed a PXN deadlock caused by an unnecessary blocking `cudaHostRegister` of legacy shared memory in the proxy process.
- Fixed single-node NVLS AllGather and ReduceScatter returning data in the wrong rank order when `CUDA_VISIBLE_DEVICES` permutes GPUs (Github PR #2377), (Issue #1906).
- Fixed a silent hang with network-registered buffers where the receiving proxy erased a rendezvous flag already published by the GPU (Github PR #2390), (Issue #2389).
- Fixed a NET/IB hang when the request counter wraps at UINT32_MAX and completions are matched to the wrong receive request.
- Fixed races in the NET/Socket transport, including a task-slot publication race that caused NULL data sends on aarch64, and a missing load ordering on the NET/IB CTS FIFO (Github PR #2365), (Github PR #2393), (Github Issue #1983).
- Fixed IB async-event threads handling events against the wrong device when `NCCL_IB_DEVICE_PCI_ORDER` reorders devices.
- Fixed communicator initialization failures on hosts with multi-port NICs when NCCL loads multiple network plugins.
- Fixed the example tuner's configuration optimizer selecting invalid measurements (Github PR #2453).
- Fixed CX9 NIC detection that misclassified BlueField-4 NICs as CX9 and missed CX9 virtual functions, preventing automatic rail and plane assignment.
- Fixed communicator initialization failures when combined per-rank topology data exceeds the topology XML fusion capacity (Github PR #2386), (Issue #2160).
- Fixed `ncclAllReduce` with `ncclAvg` on symmetric-memory buffers failing with an unhandled CUDA error for floating-point types (Github PR #2448), (Issue #2447).
- Fixed a non-blocking communicator becoming permanently unusable after one call returned `ncclInvalidArgument` (Github PR #2460), (Issue #2459).
- Fixed `ncclCommShrink` putting the parent communicator into an error state when the excluded rank list is invalid; it now returns `ncclInvalidArgument` (Github PR #2457).

- Fixed a crash in `ncclCommInitRank` when `NCCL_ALGO`, `NCCL_PROTO`, or `NCCL_SYM_KERNEL` contain empty entries (Github PR #2456), (Issue #2455).
- Fixed per-call maxCTAs bounds not being honored by kernel-based AlltoAll, Gather, and Scatter, and P2P channel counts exceeding maxCTAs after rounding to a power of two (Github PR #2435), (Issue #2421), (Issue #1598).
- Fixed `ncclCommRegister` returning success without a handle for CPU-backed buffers; it now returns a NULL handle and rejects a NULL handle pointer (Github PR #2433).
- Fixed ranks on distinct MIG compute instances of the same GPU being rejected as duplicate GPU ranks (Github Issue #1716).
- Fixed failures when `cuMem` host memory support is inconsistent across ranks. The enablement logic is no longer affected by CUDA forward compatibility.
- Fixed GDRCopy initialization failures on GPUs that support DMA-BUF export but not CPU mmap of DMA-BUF allocations.
- Fixed intermittent encrypted bootstrap failures caused by unread TLS session tickets.
- Fixed a hang or illegal memory access when a child communicator split from a `splitShare=1` parent reuses NVLS buffers in a different rank order.
- Fixed GIN and GIN-based symmetric kernels sending puts and signals to the wrong peer on communicators created with `splitShare=1`.
- Fixed CFT counted writes on the multicast path silently skipping memory binding, which could lead to hangs.
- Fixed a compiler error when calling put, putValue, or signal on a single-backend `ncclGin_BackendOne` or non-default `ncclGin_BackendMask` object (Github PR #2436).
- Fixed out-of-place AllGather performance on symmetric memory when the per-rank size is not a multiple of 256 bytes.
- Fixed a TRACE-only crash during tuning initialization for symmetric cost-model entries (Github Issue #2380).
- Fixed profiler kernel-channel completion timestamps being recorded before all warp groups finished (Github PR #2473).
- Fixed a use-after-free in RAS peer updates and a null dereference in RAS client collective handling (Github PR #2406).
- Fixed a command-injection vulnerability in the `ib_write_bw diagnostics` check by sanitizing host and device names.
- Fixed nccl4py cute GIN kernel compilation failures on CUDA 12, and CUDA compilation detection in headers when compiling with Clang (Github PR #2392), (Issue #1738).
- Fixed memory, file-descriptor, and resource leaks on various error paths (Github PR #2267), (Github PR #2364), (Github PR #2373), (Github PR #2376), (Github PR #2403), (Github PR #2404), (Github PR #2432), (Github Issue #2189).

Known Issues

- Compiling NCCL from source targeting PTX can fail due to CUDA device runtime library conflicts during linking. Directly targeting SASS does not have the same issue. This will be fixed in the next release.

Updating the GPG Repository Key

To best ensure the security and reliability of our RPM and Debian package repositories, NVIDIA is updating and rotating the signing keys used by apt, dnf/yum, and zypper package managers beginning on April 27, 2022. Failure to update your repository signing keys will result in package management errors when attempting to access or install NCCL packages. To ensure continued access to the latest NCCL release, please follow the updated NCCL installation guide.

Chapter 3. NCCL Release 2.32.3

This is the NCCL 2.32.3 release notes. For previous NCCL release notes, refer to the NCCL Archives.

Compatibility

NCCL 2.32.3 has been tested with the following:

- Deep learning framework containers. Refer to the Support Matrix for the supported container version.
- This NCCL release supports CUDA 12.x and CUDA 13.x.

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- Vera Rubin Support: Added initial Rubin platform support including support for sm107, CX9 rail and plane detection, and MPS+MLoPart. NCCL 2.32.3 focuses on new functionalities and does not contain performance model tuning for Rubin. This will be part of the next release to improve out-of-box experience for Rubin users.
- Device API Enhancements: Added Compute Fabric Transport (CFT) counted-write and wait support. Added socket-based GIN support, enabling custom kernel development with TCP sockets. Added support in GDAKI for LAG-aware QP assignment based on the context ID (Github PR #2315). Added `NCCL_WIN_GIN_ONLY` so users can register a window only for GIN usage. Optimized GIN performance by skipping mcst operation when possible.
- Collectives and Runtime Enhancements: Added a ring-based hierarchical copy-engine AllGather implementation selectable with `NCCL_HIER_CE_COLL_AG_RAIL_RING_ENABLE` (Github PR #2299). Improved Blackwell symmetric AllGather performance, resource overhead modeling and kernel selection with a new cost model. Added optional TLS encryption for NCCL-owned socket traffic when NCCL is built with OpenSSL3, configured through the `ncclSetEncryption` API. Added `ncclCollConfig_t::launchCompletionEvent`, allowing callers to observe kernel-launch completion. Added `ncclNvlsHostMode_t` to `ncclConfig_t` so applications can disable host NVLS collectives per communicator. Optimized NVLS slot consumption to avoid resource exhaustion issues when using multiple communicators with NVLS.

- Diagnostics and Profiling: Added the ATTN log level for important non-fatal conditions, such as configuration fallbacks and plugin initialization failures. Expanded RAS capability with GPU-resident progress counters and a watchdog DMA mirror for diagnosing stalled collective kernels. Added additional RAS diagnostics functionalities for NVLink/NIC state and speed, PCI and GDR configuration, Xid/SXid events and others. Reported degraded NVLink fabric bandwidth with an ATTN message during communicator initialization. Reported mismatched NCCL Git revisions across communicator ranks during initialization.
- Other Improvements: Fixed PAT+NVLS performance drops on H100 platforms. Avoided address space exhaustion when repeatedly registering symmetric windows backed by a single physical memory allocation. Reduced communicator initialization overhead at large scale by avoiding scans of inactive proxy poll descriptors. Added an experimental built-in NetworkDirect transport on Windows, with automatic socket fallback. EFA team improved EFA GDA support with gin.get API and others (Github PR #2382)(Github PR #2398)(Github PR #2405). Reported Inspector ring-buffer drops and operation counts in JSON (Github PR #2304). NCCL now supports communicators using multiple MIG instances. Generated llms.txt for agents to better read NCCL documentation.
- Contrib/Updates: Added more functionality in NiiN. Fixed contrib/nccl_checkpoint rejecting otherwise compatible NCCL versions (Github Issue #2347).

Fixed Issues

The following issues have been resolved in NCCL 2.32.3:

- Fixed profiler overhead when no profiler plugin is loaded (Github Issue #2355).
- Fixed P2P IPC registration reuse producing out-of-bounds remote addresses when a registered allocation spans multiple cuMem segments (Github PR #2362).
- Fixed PAT connection-setup deadlocks when runtime connection is disabled and nodes have uneven local-rank counts (Github Issue #2385).
- Fixed non-thread-safe token parsing that could corrupt concurrent configuration parsing (Github Issue #2361).
- Fixed B40 TMA symmetric kernels crash due to insufficient shared memory.
- Fixed GIN GDAKI bug related to hop limit when using DOCA SDK.
- Fixed GIN Proxy and GPI descriptor shared-memory sizing and alignment.
- Fixed CFT window registration performed before `ncclDevCommCreate`.
- Fixed profiler API events reporting rank 0 instead of the originating communicator rank (Github Issue #2300).
- Fixed tuner plugins receiving uninitialized cost-model constants after the tuning rework.

Known Issues

- Rubin performance model has not been optimized with NCCL 2.32.3. Users can observe better performance on Rubin by increasing the number of CTAs NCCL uses. Automatic tuning will be improved in the next release.
- Socket GIN currently requires users to opt-in to enable GDRCopy.
- B100 PCIe MLoPart workloads can fail during memory allocation with CUDA error 101 (invalid device ordinal). Set `NCCL_CUMEM_ENABLE=0` as a workaround.

Updating the GPG Repository Key

To best ensure the security and reliability of our RPM and Debian package repositories, NVIDIA is updating and rotating the signing keys used by apt, dnf/yum, and zypper package managers beginning on April 27, 2022. Failure to update your repository signing keys will result in package management errors when attempting to access or install NCCL packages. To ensure continued access to the latest NCCL release, please follow the updated NCCL installation guide.

Chapter 4. NCCL Release 2.31.2

This is the NCCL 2.31.2 release notes. For previous NCCL release notes, refer to the NCCL Archives.

Compatibility

NCCL 2.31.2 has been tested with the following:

- Deep learning framework containers. Refer to the Support Matrix for the supported container version.
- This NCCL release supports CUDA 12.x and CUDA 13.x.

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- Compute Fabric Transport (CFT): Added CFT host and device APIs for registering window memory with CUDA logical endpoints and issuing device-side Put, Get, Red and NVLS operations. CFT is supported on Blackwell GPUs with CUDA toolkit 13.3 or later.
- Per-Collective Configuration and Tuning: Added new `ncclCollConfig_t` and `nccl*Config` APIs for all collectives, with a vendor-defined field to make it easier for custom forks to integrate. Supports per-collective algorithm selection, CTA/CGA size and CTA policy overrides. Leverages collConfig to add userTag in profiler API, motivated by community RFE https://github.com/NVIDIA/nccl/issues/1916.
- GIN Enhancements: AWS-EFA team contributed the EFA GDA backend for GIN Put, Signal, and Flush operations (Github PR #2273). Added per-DevComm GIN backend selection, allowing users to create multiple DevComms with different backends. Added support for connecting GIN with custom strides. Added device-side timeouts to blocking GIN APIs such as Flush, Wait, WaitSignal, and WaitCounter operations. Reduced QP usage when using railed GIN. Reduced file-descriptor consumption in GDAKI when using many QPs. Optimized error reporting in GDAKI via event-based CQ error reporting. Added support for out-of-order delivery (DDP) in GDAKI on SPCX, improving performance without impacting GIN correctness guarantees. Added GRH support and automatic path-MTU discovery to the GIN GDAKI backend.

- Parallel Aggregated Tree (PAT) Enhancements: Enhanced the PAT algorithm for ReduceScatter/AllGather by adding hierarchical kernels that use NVLS within the node and PAT across nodes. Enables better small/medium message performance. Disabled by default. Use `NCCL_ALGO=PAT`, or the collConfig API to enable.
- One-Sided RMA and Copy Engine Collectives: Added support for multiple contexts and multiple signals to one-sided RMA operations, allowing one-sided RMA traffic from a single rank to use multiple NICs. Leverage multiple contexts and signals for hierarchical 0-SM AllGather and AllToAll. Optimizes small message latency by using NVLink multicast for AllGather.
- Tuning and Cost Model: Introduced a new unified cost model interface for querying the cost estimate for both legacy and device-API-based kernels. On Blackwell, TMA kernels are now enabled by default and integrated in the cost model for symmetric registered memory.
- Diagnostics and Profiling: Added NCCL RAS Diagnostics checks through `NCCL_RUN_RAS_DIAGNOSTICS=1` or RAS client, including GPU inventory, CUDA driver versions, ECC errors, NVLink state, and `NCCL_*` environment consistency checks. Added NCCL Diagnostics through `NCCL_RUN_DIAGNOSTICS=1`, including active P2P connectivity check and actionable P2P remediation guidance. Moved Kernel-channel profiling to a dedicated per-communicator thread, extending kernel timing to proxy-less NVLink/SHM and graph-captured collectives. Profiler V7 exposes per-kernel initial_sync, compute and final_sync phase events and symmetric-kernel variant metadata. Added per-QP CPU WQE post-to-poll latency monitoring for the IB transport.
- Added backward compatibility support for applications that JIT-compile NCCL Device API code.
- Added support for LTO IR for NCCL device API.
- Reduced communicator host-memory use and topology initialization time on large systems by allocating topology paths according to their actual lengths.
- Added multiple GIN proxy progress threads with per-thread endpoint assignment through `GIN_PROXY_NTHREADS` (Github PR #2279).
- Improved GIN host-proxy throughput by processing multiple GIN operations per progress iteration (Github PR #2232) and adding hints to the RMA plugin to aggregate requests (Github PR #2254).
- Added CuTeDSL bindings for GIN Get, Flush, Signal, ReadSignal, and PutValue operations (Github PR #2266).
- Added an internal CUDA 13.3 DMA-BUF mmap backend for GDRCopy.
- Added event-based load balancing for the net_ib transport.
- Refreshed local GIDs after `IBV_EVENT_GID_CHANGE` events during IB port recovery.
- Updated the RMA plugin interface to v15 (Github PR #2254).
- Improved AMD EPYC topology modeling (Github PR #2036).

- Improved algorithm selection on newer Intel CPUs.
- Made `nccl_device.h` compatible with C99 host translation units.
- Contrib updates: Added PACE, a parallelism-aware collective engine that fuses layout conversion, data-type conversion, and scatter/gather operations into collective kernels (Github PR #2319). Added experimental `nccl4rust` host and Device API bindings using LTO IR. Added NIIN, a header-only NVSHMEM-compatible interface implemented with NCCL Device API primitives. Added community-contributed architecture learning guides for communicator initialization, topology, tuning, transports, and collective execution (Github PR #2081). Added a community-contributed analysis of DeepEPv2.

Fixed Issues

The following issues have been resolved in NCCL 2.31.2:

- Fixed a hang when PXN connection initialization races with communicator teardown.
- Fixed data corruption when multiple symmetric windows are carved from the same backing memory allocation (Github Issue #2198).
- Fixed a hang during the first Copy Engine collective on a two-rank communicator (Github Issue #2241).
- Fixed NIC/GPU assignment perf regressions by limiting consistent start-NIC selection to Blackwell systems with ConnectX-8.
- Fixed missing GIN signal and counter requirements when Device API communicators are created asynchronously (Github PR #2208).
- Fixed a memory leak when Device API communicator creation fails (Github PR #2225).
- Fixed an incorrect GIN context count reported to applications when the allocated context count is rounded up (Github PR #2301).
- Fixed a resource leak when GIN connection setup fails (Github PR #2206).
- Fixed the CUDA thread capture mode remaining changed when GIN setup fails (Github PR #2229).
- Fixed inconsistent NVLS enablement when multiple ranks share a GPU (Github PR #2257).
- Fixed valid communicator configurations being rejected when only one of `minCTAs` or `maxCTAs` is set (Github PR #2256).
- Fixed communicator initialization failures on multi-system and MNNVL topologies when `NCCL_P2P_PXN_LEVEL=1` (Github PR #2258).
- Fixed a Device API `loadConst` performance regression caused by `__ldg`.
- Fixed CuTeDSL GIN barrier initialization failures caused by under-aligned by-value arguments (Github PR #2243, Issue #2242).

Known Issues

- PAT + NVLS has performance regressions on H100 platforms. This will be fixed in the next release.
- B40 symmetric TMA kernels: On B40 GPUs, TMA-based symmetric kernels can exceed available shared memory and fail with an illegal memory access. Set `NCCL_SYM_TMA_ENABLE=0` to disable these kernels.
- B100 PCIe MLoPart: Multi-rank-per-GPU workloads using MPS MLoPart on B100 PCIe systems can fail during memory allocation with CUDA error 101 (invalid device ordinal). Set `NCCL_CUMEM_ENABLE=0` as a workaround.

Updating the GPG Repository Key

To best ensure the security and reliability of our RPM and Debian package repositories, NVIDIA is updating and rotating the signing keys used by apt, dnf/yum, and zypper package managers beginning on April 27, 2022. Failure to update your repository signing keys will result in package management errors when attempting to access or install NCCL packages. To ensure continued access to the latest NCCL release, please follow the updated NCCL installation guide.

Chapter 5. NCCL Release 2.30.7

This is the NCCL 2.30.7 release notes. For previous NCCL release notes, refer to the NCCL Archives.

Compatibility

NCCL 2.30.7 has been tested with the following:

- Deep learning framework containers. Refer to the Support Matrix for the supported container version.
- This NCCL release supports CUDA 12.x and CUDA 13.x.

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- Zero-SM Collectives: Added hierarchical zero-SM collectives, AllGather and All2all, that use the RMA CPU proxy for inter-node communication and Copy Engines for intra-node communication. Enables better overlap of compute and communication. Enable hierarchical zero-SM collectives with the `NCCL_CTA_POLICY_ZERO` flag.
- GIN Enhancements: Added a new experimental GPU Push Interface (GPI) backend for GIN. Added explicit signal semantics with Strong and Weak signals. Added proper ncclGinFenceLevel semantics for barriers. Added a separate `NCCL_GIN_IB_TC` toggle to control traffic class used by GIN. Added `NCCL_GIN_RESOURCE_SHARING_THREAD` to enable more optimizations. Optimized QP overhead, including GDAKI mode when counters were not used. Ensured GIN was usable when NIC fusion was enabled. Added a GIN plugin example in `plugins/gin/example`.
- Symmetric Memory Improvements: Restructured the RMA plugin architecture. Added support for asymmetric buffer sizes during window registration. Optimized ReduceScatter symmetric kernel performance. Optimized performance for RMA operations using CE. Added batched CE operations to improve performance in the RMA CE put/wait path. Added support for window registration during CUDA graph capture.
- MPS with MLOPart Support (Experimental): Leveraged the CUDA Memory Locality Optimized Partition (MLOPart) feature. Supported up to 2 ranks per physical GPU with MPS and MLOPart.

- Added support for IB ports that required global route headers (GRH).
- Added logic to gin.flush to ensure all prior gets are visible.
- Added makefile support to compile Python wheels from source.
- Added `NCCL_RMA_DISABLE` environment variable to enable or disable RMA (Github PR #2151).
- Implemented reset-without-zeroing for signals and counters in GIN (Github PR #2155).
- Pin GIN proxy thread to NUMA-local CPU set (Github PR #2182).
- Added optimized weight transfer APIs in `contrib/nccl_xfer`.
- Added custom kernels in `contrib/custom_algos` for alltoall and allreduce using NCCL Device API.
- Added examples of Root Mean Square Normalization (RMSNorm), demonstrating the fusion of computation and communication using the device API.
- Unified coding style by using clang-format. See `docs/dev_guide/nccl_coding_style.md` for more details.
- Dropped support for v11 and v12 GIN plugin APIs.

Fixed Issues

The following issues have been resolved in NCCL 2.30.7:

- Fixed a deadlock caused by cuda stream allocation under PXN when memseting a buffer at runtime.
- Reintroduced cudaGridDependencySynchronize in built-in symmetric kernels, ensuring that newly launched kernels cannot access memory modified by prior kernels before it reaches point of coherency.
- Ignored system headers in include/header processing, thereby avoiding excessive realpath calls in some builds (Github PR #1806).
- Improved QP load balancing on systems configured with RoCE LAG with the round-robin queue affinity policy (Github PR #2150).
- Fixed issue when receiving an external TCP request causes the proxy thread's ncclProxyService to hang (Github PR #1834).
- Fixed `rma_proxy` MR registration type for host-NUMA `cpuAccessSignals`, which ensures that the net plugin does not reject the registration due to wrong memory type (Github PR #2187).
- Fixed GIN init context leak (Github PR #2179).
- Fixed issue with one-sided host APIs when a custom GIN plugin is used.
- Fixed one-sided host API issue where requests are dropped at a high message rate (Github Issue #2119).

Known Issues

- NCCL one-sided host RMA APIs, e.g., `ncclPutSignal`, require every rank to call the API as a one-time initialization warm-up. This will be fixed in an upcoming release.

Updating the GPG Repository Key

To best ensure the security and reliability of our RPM and Debian package repositories, NVIDIA is updating and rotating the signing keys used by apt, dnf/yum, and zypper package managers beginning on April 27, 2022. Failure to update your repository signing keys will result in package management errors when attempting to access or install NCCL packages. To ensure continued access to the latest NCCL release, please follow the updated NCCL installation guide.

Chapter 6. NCCL Release 2.30.4

This is the NCCL 2.30.4 release notes. For previous NCCL release notes, refer to the NCCL Archives.

Compatibility

NCCL 2.30.4 has been tested with the following:

- Deep learning framework containers. Refer to the Support Matrix for the supported container version.
- This NCCL release supports CUDA 12.x and CUDA 13.x.

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- Added support for elastic buffers with GIN.

Fixed Issues

The following issues have been resolved in NCCL 2.30.4:

- Fixed issue with extra headers for nccl_param feature (Github issue #2105).

Known Issues

- Applications that use GIN APIs need to be recompiled with 2.30.4 to work with 2.30.4 runtime.
- `gin.get` requires GDAKI and is not supported on Ampere or directNIC platforms.
- The elastic buffer feature with GIN is not supported on directNIC platforms.

Updating the GPG Repository Key

To best ensure the security and reliability of our RPM and Debian package repositories, NVIDIA is updating and rotating the signing keys used by apt, dnf/yum, and zypper package managers beginning on April 27, 2022. Failure to update your repository signing keys will result in package management errors when attempting to access or install

NCCL packages. To ensure continued access to the latest NCCL release, please follow the updated NCCL installation guide.

Chapter 7. NCCL Release 2.30.3

This is the NCCL 2.30.3 release notes. For previous NCCL release notes, refer to the NCCL Archives.

Compatibility

NCCL 2.30.3 has been tested with the following:

- Deep learning framework containers. Refer to the Support Matrix for the supported container version.
- This NCCL release supports CUDA 12.x and CUDA 13.x.

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- Device API and GIN Enhancements: GIN contexts are no longer shared between device communicators backed by the same host communicator. Added per-context resource sharing modes for GIN, allowing GPU-scope or CTA-scoped resource. Added TrafficClass support to device Added versioning to `ncclDevComm`. Added timeout support to the device APIs. Added `max_rd_atomic` and `max_dest_rd_atomic support` in GIN. Upgraded doca-gpunetio to v2.0.2-rc1.
- Elastic Buffers (LSA): Introduced feature to support new use cases where large tensors are split into multi-segment windows, with the active region in GPU memory and the remainder in host memory. Enables larger effective models and reduces memory pressure during spilling. Elastic buffers will support GIN in a future release.
- `gin.get` with Nonblocking Flush: Added support for GPU-initiated gets and checking completion without stalling. Currently only works with GDAKI (not with CPU proxy) and doesn't work on directNIC and Ampere.
- Symmetric Memory: Added AVG operator to ReduceScatter Symmetric kernels. Enabled dynamic memory offload with group support for single-process, multi-GPU scenarios. Added support for GPU-only multi-segment registration for symmetric windows. Added CUDA graph capture and replay support for `ncclPutSignal` and `ncclWaitSignal` APIs. One-sided RMA can now use an external network plugin.

- Tensor Memory Accelerator (TMA): Added TMA support in select built-in symmetric kernels to offload bulk peer-to-peer copies and reductions, improving NVLink bandwidth and latency. Can be enabled with `NCCL_SYM_TMA_ENABLE=1`.
- DDP Support : Enables Dynamic Direct Path (DDP) so that NCCL can take advantage of hardware multipath and out-of-order receive for higher network performance on supported systems. Can be enabled with `NCCL_IB_OOO_RQ=1`.
- Port Recovery: Added support for IB port recovery in NCCL. Improved NCCL's ability to recover from transient network issues so communicators can continue operating without full re-initialization. Can be enabled with `NCCL_IB_RESILIENCY_PORT_RECOVERY=1`.
- Cross Clique Support: Added support for treating multiple cliques as the same NVLINK domain. Can be enabled with `NCCL_MNNVL_CROSS_CLIQUE=1`.
- NCCL Parameter Infrastructure: Added new C APIs to support querying NCCL parameters. Introduced `ncclParamGetAllParameterKeys`, `ncclParamDumpAll`, `ncclParamGet` and `ncclParamGetParameter` APIs.
- NCCL4PY v0.2.0: Added new APIs from NCCL 2.29 release. Added devcomm create/destroy APIs to prepare for device API. Enabled Freethreading support. Added NCCL Inspector P2P event support.
- `ncclGinBarrierSession` can now be created directly for the world team without manual resource allocation.
- GIN proxy GFD size increased to 128 bytes with version field added.
- GIN proxy CQ polling (`ginProgress`) moved to per-context to improve performance.
- `ncclBarrierSession` no longer shares resources with `ncclLsaBarrierSession` or `ncclGinBarrierSession`.
- Redundant `NCCL_DEBUG=INFO` log volume reduced significantly.
- Added NVLSTree tuning that improves performance for various Blackwell systems.
- Added p2pMaxPeers to communicator to achieve better tuning for send/recv vs. all2all.
- Enabled LL128 protocol in heterogeneous scenarios for Hopper and later GPUs.
- Added checks for mismatched Net and CollNet counts across communicators.
- Added Graphana template for NCCL inspector dashboard rendering using Prometheus data.
- Removed unused members `nccl_id`, `comm`, `nccl_unique_id`, and `thread_ranks` in the examples (Github PR #1989).
- Added `NCCL_LIBIBVERBS_SO` environment variable to specify an absolute path for libibverbs (Github PR #2043).
- Extended suspend memory offload to channel device allocations (Github PR #2060).

Fixed Issues

The following issues have been resolved in NCCL 2.30.3:

- Fixed implicit CUDA synchronization in `putSignal` and `CE collectives` caused by pageable CPU stack memcpy.
- Fixed a hang when using CE collectives and cuda graph under an edge case.
- Fixed NULL access issue during finalize when RMA and GIN plugins are both initialized.
- Fixed race conditions in all2all GIN/Hybrid examples with more than one CTA.
- Fixed `ncclGinType_t uint8_t` enum compatibility issue in `nccl4py`.
- Fixed several memory leaks in communicator create/destroy code paths.
- Fixed a bug in plugin compat layer for v11 related to lazy initialization.
- Fixed data corruption in symmetric LL kernels with unaligned buffer.
- Fixed plugin name being cleared after communicator destroy (Github Issue #1978).
- Fixed deadlock and use-after-free in the inspector plugin (Github Issue #2000).
- Fixed incorrect network interface selection caused by inverted Boolean logic in `matchSubnet` (Github PR #2047).
- Fixed regression from 2.29.2 where CPU affinity mask is not restored in `initTransportsRank` (Github issue #2033).

Known Issues

- Applications that use GIN APIs need to be recompiled with 2.30.3 to work with 2.30.3 runtime.
- `gin.get` requires GDAKI and is not supported on Ampere or directNIC platforms.

Updating the GPG Repository Key

To best ensure the security and reliability of our RPM and Debian package repositories, NVIDIA is updating and rotating the signing keys used by apt, dnf/yum, and zypper package managers beginning on April 27, 2022. Failure to update your repository signing keys will result in package management errors when attempting to access or install NCCL packages. To ensure continued access to the latest NCCL release, please follow the updated NCCL installation guide.

Chapter 8. NCCL Release 2.29.7

This is the NCCL 2.29.7 release notes. For previous NCCL release notes, refer to the NCCL Archives.

Compatibility

NCCL 2.29.7 has been tested with the following:

- Deep learning framework containers. Refer to the Support Matrix for the supported container version.
- This NCCL release supports CUDA 12.x and CUDA 13.x.

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- Device API and GIN: Added multi-context support for GIN with the option to request exclusive GIN contexts. Added VA-based GIN signals plus strict window ordering. Added advanced queue control for GIN, including queue depth, manual credit management and aggregation. Added GIN support for platforms with no cross rail connectivity. Added `nLsaTeams` to `ncclCommQueryProperties`. Decoupled GIN from NET plugin and topology.
- New device APIs: Introduced Copy, ReduceCopy, and ReduceSum with various data types and ops.
- Dynamic Memory Offload: Added `ncclCommSuspend()` and `ncclCommResume()` for releasing and restoring communicator memory. Added basic memory overhead tracking infrastructure. Multiple GPUs for a single process are not currently supported for `ncclCommSuspend()` and `ncclCommResume()`, this will be added in v2.30.
- Built-in hybrid (LSA+GIN) symmetric kernel for ReduceScatter: Added new hierarchical kernels to improve performance and scalability of ReduceScatter. Requires symmetric memory registration and GIN support. Symmetric GIN kernels can be disabled with `NCCL_SYM_GIN_KERNELS_ENABLE=0`.
- Port Failover: Added support for internal IB/RoCE plugin to continue working transparently when network errors occur. Added automatic port failover for

GPUs having multiple local IB/RoCE ports/devices. Can be enabled by setting `NCCL_IB_RESILIENCY_PORT_FAILOVER=1`.

- Symmetric memory: Added support for abort in symmetric kernels. Added `NCCL_CHECK_MODE=DEBUG` to validate symmetric buffer registration.
- Project layout reorganization: Moved the ext-* directories to plugins (e.g., ext-net -> plugins/net). Moved ir and nccl4py under bindings. Moved examples to docs/examples.
- Uses different signals for different peers in the GIN barrier.
- Added `NCCL_NO_CACHE` to force NCCL to always re-read selected env vars.
- Added CMake install and find_package support.
- Added CMake for NCCL4Py build and updated Cybind integration.
- Added preliminary backwards compatibility support to enable running LSA kernels compiled with NCCL 2.29.2/3 on NCCL 2.29.7. This is not supported for GIN yet.
- Updated license to Apache 2.0.

Fixed Issues

The following issues have been resolved in NCCL 2.29.7:

- Fixed problems related to the introduction of git_version.h. (Github Issue #1960)
- Fixed oneRankReduce when the number of elements is not a multiple of block number. (Github Issue #1950)
- Improved GIN handling in `ncclCommGetAsyncError`. (Github Issue #2019)
- Fixed memory initialization in P2P transport. (Github Issue #1962)
- Fixed hang issue in send/receive scheduling of repeated sparse patterns.
- Fixed CE-based collective operations to fall back to cudaMemcpyAsync API when null/default stream is used.
- Fixed symmetric window objects to be freed automatically during `commFree`.
- Fixed a 16-bit overflow of signal and counter ids with GIN proxy.
- Fixed GIN counters and signals to be reset upon `ncclDevCommDestroy`.
- Fixed local data calculation during `ncclGinIbP2PBarrier`.
- Fixed a crash when calling ncclNet.finalize() after a failed ncclNet_v10->init().
- Fixed nccl4py compatibility with cuda.core 0.5.0.

Known Issues

- Applications that use GIN APIs need to be recompiled with 2.29.7 to work with 2.29.7 runtime.
- The Profiler Inspector example does not currently compile under CMake. This will be fixed soon.

Updating the GPG Repository Key

To best ensure the security and reliability of our RPM and Debian package repositories, NVIDIA is updating and rotating the signing keys used by apt, dnf/yum, and zypper package managers beginning on April 27, 2022. Failure to update your repository signing keys will result in package management errors when attempting to access or install NCCL packages. To ensure continued access to the latest NCCL release, please follow the updated NCCL installation guide.

Chapter 9. NCCL Release 2.29.3

This is the NCCL 2.29.3 release notes. For previous NCCL release notes, refer to the NCCL Archives.

Compatibility

NCCL 2.29.3 has been tested with the following:

- Deep learning framework containers. Refer to the Support Matrix for the supported container version.
- This NCCL release supports CUDA 12.x and CUDA 13.x.

Fixed Issues

The following issues have been resolved in NCCL 2.29.3:

- Fixed CAS usage in case of weak failure which was causing a hang on ARM. The issue affected NCCL when compiled with gcc versions prior to 10.

Updating the GPG Repository Key

To best ensure the security and reliability of our RPM and Debian package repositories, NVIDIA is updating and rotating the signing keys used by apt, dnf/yum, and zypper package managers beginning on April 27, 2022. Failure to update your repository signing keys will result in package management errors when attempting to access or install NCCL packages. To ensure continued access to the latest NCCL release, please follow the updated NCCL installation guide.

Chapter 10. NCCL Release 2.29.2

This is the NCCL 2.29.2 release notes. For previous NCCL release notes, refer to the NCCL Archives.

Compatibility

NCCL 2.29.2 has been tested with the following:

- Deep learning framework containers. Refer to the Support Matrix for the supported container version.
- This NCCL release supports CUDA 12.x and CUDA 13.x.

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- Device API has improved GIN documentation to clarify the support matrix, added host-accessible device pointer functions from symmetric registered ncclWindows, added a new ncclCommQueryProperties API to check supported features before creating a DevComm, and modified structs to add versioning for backwards compatibility in future versions.
- One-sided Host APIs were added (put, wait) for both network and NVL using zero-SM. Requires CUDA 12.5 or greater.
- Added NCCL4Py, Python language bindings for NCCL. Supports native collectives, P2P and other NCCL operations, with automatic cleanup of NCCL-managed resources.
- Added LLVM intermediate representation (IR) support, enabling consumption by diverse code generation systems.
- Added a built-in hybrid (LSA+GIN) symmetric kernel for AllGather.
- Implemented the new ncclCommGrow API which adds the ability to dynamically and efficiently add ranks to an existing NCCL communicator. Use it with ncclCommShrink to adjust membership of communicators in response to failing and recovering nodes. It also addresses the need for elastic applications to expand a running job by integrating new ranks.

- Added multi-segment restoration which expands buffer registration to support multiple segments of physical memory mapped to one contiguous VA space for the p2p, ib, and nvls transports. This enabled support for expandable segments in PyTorch.
- Improved scalability of AllGatherV with a new scalable allgatherv pattern (group of broadcasts) and a new scheduler path and new kernels to improve performance at large scale.
- Improved debuggability and observability with realtime monitoring support in RAS, Prometheus output format for Inspector, and profiler support for CopyEngine (CE) based collectives.
- Added a new contribution guide available at: https://github.com/NVIDIA/nccl/blob/master/CONTRIBUTING.md
- Added `NCCL_SOCKET_POLL_TIMEOUT_MSEC` which allows waiting instead of spinning during bootstrap in order to reduce CPU usage. (Github PR #1759)

Fixed Issues

The following issues have been resolved in NCCL 2.29.2:

- Fixed segfault in ncclGin initialization that can happen if `ncclGinIbGdaki.devices()` fails after `init()` succeeds. (Github PR #1881)
- Fixed crash that can happen when calling p2p and then collectives while using the same user buffer. (Github Issue #1859)
- Fixed bug that was lowering performance on some sm80 or earlier machines with one NIC per GPU. (Github Issue #1876)
- Cleared non-fatal CUDA errors so they do not propagate. (PyTorch Issue #164402)
- Improved performance of large-size AllGather operations using symmetric memory buffers on Blackwell by transparently switching to CE collectives.
- Improved the default number of channels per net peer for all-to-all, send, and recv to achieve better performance.
- Improved performance tuning of 256M-512M message sizes on Blackwell for AllReduce.
- Enabled built-in symmetric kernels only on fully connected nvlink systems, as PCIE systems do not perform as well.
- Updated all2all, send, and recv to obey `NCCL_NETDEVS_POLICY`. For these operations, NCCL will by default use a subset of available network devices as dictated by the Network Device Policy.
- Fixed a hang on GB200/300 + CX8 when the user disables GDR.
- Fixed a bug that could cause AllReduce on ncclFloat8e4m3 to yield `no algorithm/protocol available`.

Updating the GPG Repository Key

To best ensure the security and reliability of our RPM and Debian package repositories, NVIDIA is updating and rotating the signing keys used by apt, dnf/yum, and zypper package managers beginning on April 27, 2022. Failure to update your repository signing keys will result in package management errors when attempting to access or install NCCL packages. To ensure continued access to the latest NCCL release, please follow the updated NCCL installation guide.

Chapter 11. NCCL Release 2.28.9

This is the NCCL 2.28.9 release notes. For previous NCCL release notes, refer to the NCCL Archives.

Compatibility

NCCL 2.28.9 has been tested with the following:

- Deep learning framework containers. Refer to the Support Matrix for the supported container version.
- This NCCL release supports CUDA 12.x and CUDA 13.x.

Fixed Issues

The following issues have been resolved in NCCL 2.28.9:

- Fixed operation ordering between main thread and proxy thread to prevent hangs at large scale.
- Fixed Issue #1893, a bug fix in GIN.

Updating the GPG Repository Key

To best ensure the security and reliability of our RPM and Debian package repositories, NVIDIA is updating and rotating the signing keys used by apt, dnf/yum, and zypper package managers beginning on April 27, 2022. Failure to update your repository signing keys will result in package management errors when attempting to access or install NCCL packages. To ensure continued access to the latest NCCL release, please follow the updated NCCL installation guide.

Chapter 12. NCCL Release 2.28.7

This is the NCCL 2.28.7 release notes. For previous NCCL release notes, refer to the NCCL Archives.

Compatibility

NCCL 2.28.7 has been tested with the following:

- Deep learning framework containers. Refer to the Support Matrix for the supported container version.
- This NCCL release supports CUDA 12.x and CUDA 13.x.

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- Introduced GPU-Initiated Networking (GIN), an experimental device-side API to extend the Device API to support communication of the network. The signatures and functionality may evolve in future releases. Applications using the device APIs must be recompiled with new NCCL releases. The support for ABI compatibility will be added in the future.
- Improved the Device API with a new ncclBarrierSession feature; removing a restriction on multimem to make it available to as few as two ranks; and removing the distance (`NCCL_P2P_LEVEL`) considerations from determining the availability of symmetric memory.
- Added ncclCommRevoke API to enhance fault tolerance by enabling NCCL to quiesce ongoing work on a communicator without freeing resources.
- Introduced a new NCCL Environment Plugin that allows users to set NCCL parameters without having to set environment variables for the process.
- Released a set of examples that highlight NCCL's core features, including communicator initialization, point-to-point communication, and collective operations. Advanced features like user buffer registration, symmetric memory, and the device API are also covered.
- Enhanced NCCL RAS output to add JSON in order to support machine-parsable metrics collection.

- Integrated CPU Optimizations for NCCL Initialization at Large Scale.
- Improved Bootstrap AllGather by 2x at large scale by sending bootstrap information bidirectionally.

Fixed Issues

The following issues have been resolved in NCCL 2.28.7:

- Fixed multicast object leaks in case of failed NVLS user buffer registrations.
- Fixed potential data corruption with built-in symmetric kernels for small messages with size granularity under 8 bytes or when multiple symmetric operations were aggregated in a group.
- Generalized the existing point-to-point scheduling to the case of un-even GPU count per node.
- Fixed a crash when network plugin assignment fails.
- Fixed a large performance issue with `NCCL_CROSS_NIC=0` and certain split mask settings, where NCCL could not find a viable ring.
- Fixed crash when NCCL was compiled with recent CUDA versions but run on hosts with certain specific older CUDA drivers.
- Fixed spurious failures when PyTorch was statically linked with NCCL-2.28.3.

Updating the GPG Repository Key

To best ensure the security and reliability of our RPM and Debian package repositories, NVIDIA is updating and rotating the signing keys used by apt, dnf/yum, and zypper package managers beginning on April 27, 2022. Failure to update your repository signing keys will result in package management errors when attempting to access or install NCCL packages. To ensure continued access to the latest NCCL release, please follow the updated NCCL installation guide.

Chapter 13. NCCL Release 2.28.3

This is the NCCL 2.28.3 release notes. For previous NCCL release notes, refer to the NCCL Archives.

Compatibility

NCCL 2.28.3 has been tested with the following:

- Deep learning framework containers. Refer to the Support Matrix for the supported container version.
- This NCCL release supports CUDA 12.x and CUDA 13.x.

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- Added Device API, an experimental device-side API to integrate NCCL communication directly into application kernels. The signatures and functionality may evolve in future releases. No ABI compatibility is currently guaranteed for this API. Applications must be recompiled with each new NCCL release.
- Symmetric kernels were reimplemented using the device API and have support for aggregating symmetric operations using ncclGroupStart/End.
- Added new host collective APIs: ncclAlltoAll, ncclScatter, and ncclGather.
- Added experimental support for CMake as an alternative to building with Makefiles. Known issues to be fixed in an upcoming release: pkg.build and Device API are not supported.
- Reduced SM utilization within a single (MN)NVL domain by using CE (Copy Engine) Collectives. Freed up more SM capacity for the application when using ncclAlltoAll, ncclGather, or ncclScatter. To enable the feature, register buffers into symmetric windows and use the `NCCL_CTA_POLICY_ZERO` flag in the communicator `config_t`.
- Decreased max CTA count from 32 to 16 on Blackwell. This improvement decreased SM overhead by 50%, but it may cause a performance drop on Blackwell. It can be overridden by setting `NCCL_MIN_CTAS=32` and `NCCL_MAX_CTAS=32`, or setting the same in the communicator config. Based on community feedback, future versions may consider different trade-offs between performance and SM overhead.

- Network Plugin added version 11 which supports per-communicator init/finalize. Now passes information about communication operations to be executed on the network end point. Added multi-request net API to help anticipate and optimize multiple send/recv requests.
- Profiler Plugin added support for API events (group, collective, and p2p) and for tracking kernel launches. Added Inspector Profiler example plugin for always-on performance monitoring and a hook to Google's CoMMA profiler on github.
- Tuner Plugin exposed NCCL tuning constants with ncclTunerConstants_v5_t and added NVL Domain Information API.
- Plugin system added support for multiple plugin types from a single shared object.
- Added new option `NCCL_MNNVL_CLIQUE_ID=-2` which will use rack serial number to partition the MNNVL clique. This will limit NVLink domains to GPUs within a single rack.
- Added `NCCL_NETDEVS_POLICY` to control how NET devices are assigned to GPUs. The default (AUTO) is the policy used in previous versions.
- Added `NCCL_SINGLE_PROC_MEM_REG_ENABLE` control variable to enable NVLS UB registration in the "one process, multiple ranks" case as opt in.
- Moved nChannelsPerNetPeer into ncclConfig. `NCCL_NCHANNELS_PER_NET_PEER` can override the value in ncclConfig.
- Enabled PXN over C2C by default, which can improve performance for Grace-Blackwell platforms by allowing NCCL to leverage the NIC attached to a peer GPU over NVLINK, C2C, and PCIe. It can be disabled by setting `NCCL_PXN_C2C=0`.

Fixed Issues

The following issues have been resolved in NCCL 2.28.3:

- Allowed FP8 support for non-reductive operations on pre-sm90 devices.
- Fixed NVLS+CollNet and temporarily disabled `COLLNET_CHAIN` for >8 GPUs.
- Fixed considering non-running interfaces for socket traffic. Now NCCL will not attempt to use interfaces that do not have the `IFF_RUNNING` bit.
- Improved response to RoCE link flaps. Instead of reporting an "unknown event" it will now report "GID table changed."
- Moved libvirt bridge interface to the end of possible interfaces so that it is considered last. These interfaces are usually virtual bridges to relay traffic to containers running on the host and cannot be used for traffic to a remote node and are therefore unsuitable.

Updating the GPG Repository Key

To best ensure the security and reliability of our RPM and Debian package repositories, NVIDIA is updating and rotating the signing keys used by apt, dnf/yum, and zypper package managers beginning on April 27, 2022. Failure to update your repository signing

keys will result in package management errors when attempting to access or install NCCL packages. To ensure continued access to the latest NCCL release, please follow the updated NCCL installation guide.

Chapter 14. NCCL Release 2.27.7

This is the NCCL 2.27.7 release notes. For previous NCCL release notes, refer to the NCCL Archives.

Compatibility

NCCL 2.27.7 has been tested with the following:

- Deep learning framework containers. Refer to the Support Matrix for the supported container version.
- This NCCL release supports CUDA 12.2, CUDA 12.4, and CUDA 12.9. The provided prebuilt binaries should work with other CUDA 12.x versions as well. CUDA 13.0 is supported as well.

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- Added support for CUDA 13.0.

Fixed Issues

The following issues have been resolved in NCCL 2.27.7:

- Fixed initialization failures in certain configurations when attempting to load fp8-specific symmetric multicast kernels on GPUs older than Blackwell.

Known Issues

- A note for users on MNNVL systems: please ensure an adequate stack size for NCCL threads. While the default Linux stack size limit of 8192 KB is known to be sufficient, we've seen crashes if the limit is changed to "unlimited," as it causes the glibc library to unexpectedly *decrease* the stack size of NCCL's background threads to just 2048 KB. Use `ulimit -s` in bash to print the current limit; if needed, reset it to 8192 KB using `ulimit -s 8192`. (You also need to ensure that the new setting is propagated to other nodes when launching a multi-node NCCL job).

Updating the GPG Repository Key

To best ensure the security and reliability of our RPM and Debian package repositories, NVIDIA is updating and rotating the signing keys used by apt, dnf/yum, and zypper package managers beginning on April 27, 2022. Failure to update your repository signing keys will result in package management errors when attempting to access or install NCCL packages. To ensure continued access to the latest NCCL release, please follow the updated NCCL installation guide.

Chapter 15. NCCL Release 2.27.6

This is the NCCL 2.27.6 release notes. For previous NCCL release notes, refer to the NCCL Archives.

Compatibility

NCCL 2.27.6 has been tested with the following:

- Deep learning framework containers. Refer to the Support Matrix for the supported container version.
- This NCCL release supports CUDA 12.2, CUDA 12.4, and CUDA 12.9. The provided prebuilt binaries should work with other CUDA 12.x versions as well.

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- Improved support for DirectNIC (CX8): XDR speed detection and the reporting of the RDMA interfaces only when DirectNIC is enabled.
- Completed P2C (PXN over C2C) support, which is now preferred over regular PXN and extends to send/receive operations as well. This feature is currently preliminary and is disabled by default; use `NCCL_PXN_C2C=1` to enable.
- Added support for compilation with GCC 14.

Fixed Issues

The following issues have been resolved in NCCL 2.27.6:

- Fixed the unloading of network plugins that also provide tuner capability.
- Fixed the change of the current device across the calls to `ncclCommDestroy()` and `ncclCommAbort()`.

Known Issues

- A note for users on MNNVL systems: please ensure an adequate stack size for NCCL threads. While the default Linux stack size limit of 8192 KB is known to be sufficient, we've seen crashes if the limit is changed to "unlimited," as it causes the glibc library

to unexpectedly *decrease* the stack size of NCCL's background threads to just 2048 KB. Use `ulimit -s` in bash to print the current limit; if needed, reset it to 8192 KB using `ulimit -s 8192`. (You also need to ensure that the new setting is propagated to other nodes when launching a multi-node NCCL job).

Updating the GPG Repository Key

To best ensure the security and reliability of our RPM and Debian package repositories, NVIDIA is updating and rotating the signing keys used by apt, dnf/yum, and zypper package managers beginning on April 27, 2022. Failure to update your repository signing keys will result in package management errors when attempting to access or install NCCL packages. To ensure continued access to the latest NCCL release, please follow the updated NCCL installation guide.

Chapter 16. NCCL Release 2.27.5

This is the NCCL 2.27.5 release notes. For previous NCCL release notes, refer to the NCCL Archives.

Compatibility

NCCL 2.27.5 has been tested with the following:

- Deep learning framework containers. Refer to the Support Matrix for the supported container version.
- This NCCL release supports CUDA 12.2, CUDA 12.4, and CUDA 12.9. The provided prebuilt binaries should work with other CUDA 12.x versions as well.

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- Optimized the network performance on GB200 systems by alternating the direction of the rings and the NIC to GPU assignment across communicators to limit unnecessary sharing.
- Optimized the network performance on DGX B200 systems by adjusting the bandwidths provided to the graph search algorithm.
- Added an example tuner plugin with CSV-based overrides.

Fixed Issues

The following issues have been resolved in NCCL 2.27.5:

- Fixed the detection of C2C links in case GPU Direct RDMA is disabled between a GPU and a NIC.
- Fixed PXN support on MNNVL systems, where NCCL would try (and fail) to share regular host memory across multiple nodes.
- Further reduced the overheads of CUDA graph capturing, which increased in NCCL 2.26.2 for large graphs.
- Enabled fp8 reductions in symmetric kernels on Blackwell with CUDA 12.8.

- Restored the plugin name handling logic to make it possible to specify a path to the plugin.
- Restored the ability to change `NCCL_COLLNET_ENABLE` during execution
- Removed an x86 dependency from the example profiler.

Updating the GPG Repository Key

To best ensure the security and reliability of our RPM and Debian package repositories, NVIDIA is updating and rotating the signing keys used by apt, dnf/yum, and zypper package managers beginning on April 27, 2022. Failure to update your repository signing keys will result in package management errors when attempting to access or install NCCL packages. To ensure continued access to the latest NCCL release, please follow the updated NCCL installation guide.

Chapter 17. NCCL Release 2.27.3

This is the NCCL 2.27.3 release notes. For previous NCCL release notes, refer to the NCCL Archives.

Compatibility

NCCL 2.27.3 has been tested with the following:

- Deep learning framework containers. Refer to the Support Matrix for the supported container version.
- This NCCL release supports CUDA 12.2, CUDA 12.4, and CUDA 12.9. The provided prebuilt binaries should work with other CUDA 12.x versions as well.

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- Added symmetric memory API (`ncclCommWindowRegister` with the `NCCL_WIN_COLL_SYMMETRIC` flag, `ncclCommWindowDeregister`). This capability is on by default (`NCCL_WIN_ENABLE=0` can be used to disable) and depends on CUMEM being operational.
- Implemented specialized kernels taking advantage of symmetrically registered memory. Initial support includes P2P and NVLS connectivity, floating point types up to 32 bits, sum as the reduction operator, and one collective operation per group.
- Added support for DGX Spark.
- Added support for DirectNIC (CX8) to the internal IB plugin.
- Added support for communicator shrinking (ncclCommShrink).
- Added support for loading multiple network plugins (`NCCL_NET_PLUGIN` now accepts a comma-separated list of plugins to load).
- Added NVLS+IB SHARP support for AllGather and ReduceScatter with user buffer registration.
- Decreased the NVLS channel count to 24 on Blackwell systems with a single NVLink domain per communicator.

- Enabled fine-tuning of NCCL behavior per communicator using new `ncclConfig_t` members `collnetEnable`, `CTAPolicy`, and `nvlsCTAs` (the environment variable `NCCL_CTA_POLICY` can also be used to force a particular CTA policy).
- Implemented several enhancements to the profiler in terms of expanding the amount of information being provided at initialization, providing GPU-generated timestamps for the GPU kernel events, adding support for network-defined event updates, and more.
- Optimized the performance of collective operations on GB200 systems by increasing to 16 the number of NICs used to communicate between MNNVL domains.
- Added support for more complex MNNVL topologies.
- Disabled implicit fallback to a potentially much slower network transport if the MNNVL fabric initialization was unsuccessful. Such failures typically indicate a misconfigured IMEX support on the system; `NCCL_MNNVL_ENABLE=0` can be used to continue without MNNVL.
- Disabled the creation of fused NICs for physical devices that haven't been merged.
- Improved support for platforms with C2C connectivity by enabling GPUDirect RDMA for the NICs by default (`NCCL_NET_GDR_C2C=0` can be used to disable) and by adding support for P2C (PXN over C2C) and the LL128 protocol.
- Extended NCCL fault tolerance to support more complex multithreaded scenarios during initialization.
- Enabled ncclImplicitOrderLaunch for CUDA 12.9+.
- Improved the netSocket transport latency and provided finer control over, e.g., send/receive buffer size.
- Improved the readability of the CPU affinity in the debug output.

Fixed Issues

The following issues have been resolved in NCCL 2.27.3:

- Enabled graceful fallback if NVLS initialization fails (`NCCL_NVLS_ENABLE=1` can be used to preserve the old behavior of returning an error).
- Fixed several issues in the profiler support: reporting the correct number of channels used, keeping track of an event identifier, improving backward compatibility, and more.
- Fixed multiple issues related to MNNVL support: a hang in alltoall-like communication patterns at a scale of over 80 ranks, `NCCL_P2P_DISABLE=1` leaving MNNVL enabled (it now implies `NCCL_MNNVL_ENABLE=0`), an initialization failure when `NCCL_TOPO_FILE` was being used, failure to exclude non-local NICs in the graph search, and an incompatibility of the SHM transport with MNNVL.
- Fixed a potential race condition when mixing graph and non-graph execution.
- Improved PXN connection code to avoid duplicate and unused connections.

- Fixed several issues in RAS related to race conditions and memory corruption.
- Fixed a potential memory corruption in `ncclCommSplit` with resource sharing.
- Fixed a small memory leak and over-sychronization in asynchronous graph upload.
- Added a check for out-of-memory conditions in `ncclMemAlloc`.
- Cleaned up the NCCL socket code: made it more willing to retry in case of errors during connection establishment, added error checking in a few instances, and improved the debug output.
- Switched `NCCL_DEBUG_FILE` to line buffering.
- Fixed other minor issues, primarily in the graph search code and the internal IB plugin

Updating the GPG Repository Key

To best ensure the security and reliability of our RPM and Debian package repositories, NVIDIA is updating and rotating the signing keys used by apt, dnf/yum, and zypper package managers beginning on April 27, 2022. Failure to update your repository signing keys will result in package management errors when attempting to access or install NCCL packages. To ensure continued access to the latest NCCL release, please follow the updated NCCL installation guide.

Chapter 18. NCCL Release 2.26.5

This is the NCCL 2.26.5 release notes. For previous NCCL release notes, refer to the NCCL Archives.

Compatibility

NCCL 2.26.5 has been tested with the following:

- Deep learning framework containers. Refer to the Support Matrix for the supported container version.
- This NCCL release supports CUDA 12.2, CUDA 12.4, and CUDA 12.9.

Fixed Issues

The following issues have been resolved in NCCL 2.26.5:

- Minimized the performance impact of the device kernel profiling support when the profiler plugin is not loaded.
- Reduced the overheads of CUDA graph capturing, which increased in NCCL 2.26.2 for large graphs.
- Fixed the exchange of enhanced connection establishment (ECE) options to address potential slowdowns on networks utilizing RoCE.
- Added testing if cuMem host allocations work and if not, disabling them. Enabled by default since NCCL 2.24 if the CUDA driver version is at least 12.6, such allocations rely on NUMA support, which is by default not available under Docker. We recommend invoking Docker with `--cap-add SYS_NICE` to enable it.
- Worked around a potential hang in alltoall-like communication patterns on MNNVL systems at a scale of over 80 ranks.
- Fixed an initialization error when running with `NCCL_NET_GDR_C2C=1` on multiple MNNVL domains with non-uniform network configurations across nodes.
- Fixed the printing of sub-seconds in the debug log when using a custom `NCCL_DEBUG_TIMESTAMP_FORMAT` setting.

Updating the GPG Repository Key

To best ensure the security and reliability of our RPM and Debian package repositories, NVIDIA is updating and rotating the signing keys used by apt, dnf/yum, and zypper package managers beginning on April 27, 2022. Failure to update your repository signing keys will result in package management errors when attempting to access or install NCCL packages. To ensure continued access to the latest NCCL release, please follow the updated NCCL installation guide.

Chapter 19. NCCL Release 2.26.2

This is the NCCL 2.26.2 release notes. For previous NCCL release notes, refer to the NCCL Archives.

Compatibility

NCCL 2.26.2 has been tested with the following:

- Deep learning framework containers. Refer to the Support Matrix for the supported container version.
- This NCCL release supports CUDA 12.2, CUDA 12.4, and CUDA 12.8.

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- Added support for the profiling of CUDA kernel start and ends and of the network plugin. Improved profiling granularity and added support for graph capturing.
- Added implicit launch order, which allows to prevent deadlocks when using multiple NCCL communicators per device by implicitly ordering NCCL operations using the host program order. Disabled by default, set `NCCL_LAUNCH_ORDER_IMPLICIT=1` to enable.
- Added a complementary mechanism to detect host threads racing to launch to the same device. Enabled by default, set `NCCL_LAUNCH_RACE_FATAL=0` to disable.
- Significantly accelerated the PAT algorithm by separating the computation and execution of PAT steps on different warps.
- Added support for setting QoS per communicator, using a new traffic class config to allow the application to select a particular traffic class for a given communicator. For the IB/RoCE plugin, existing config variables such as `NCCL_IB_SL` and `NCCL_IB_TC` take precedence.
- Added support for enabling GPU Direct RDMA specifically on C2C platforms. Disabled by default, set `NCCL_NET_GDR_C2C=1` to enable.
- Keep user buffer registration enabled as much as possible, disabling it only when a communicator has more than one rank per node on any node.

- Report operation counts in RAS separately for each collective type and provide details about missing communicator ranks.
- Added support for timestamps to NCCL diagnostic messages. This is enabled by default for WARN messages, use `NCCL_DEBUG_TIMESTAMP_LEVELS` to specify the levels that should include a timestamp. Use `NCCL_DEBUG_TIMESTAMP_FORMAT` to adjust the format of the timestamps.
- Reduced the memory usage with NVLink SHARP (NVLS).
- Improved algorithm/protocol selection on recent Intel CPUs such as Emerald Rapids and Sapphire Rapids.
- Improved channel scheduling when mixing LL and Simple operations.
- Added support for comment lines (starting with #) in the `nccl.conf` file.
- Make user buffer registration problems print an INFO instead of a WARN.

Fixed Issues

The following issues have been resolved in NCCL 2.26.2:

- Fixed a potential hang during connection setup when multiple communicators share resources.
- Fixed a performance regression when using `NCCL_CROSS_NIC=1`.
- Made the GID index detection code more resilient to avoid issues with containers.
- Fixed a potential crash when creating a non-blocking communicator after a non-blocking collective operation on another communicator.
- Fixed shared memory usage on recent Blackwell GPUs.
- Fixed an issue where reloading the IB SHARP plugin could result in an error.
- Made the failures to auto-merge NICs non-fatal, such as when mixing IB and RoCE devices.
- Fixed a potential hang in ncclCommAbort and reduced the abort time by up to two orders of magnitude.
- Fixed a crash when `libnccl.so` was dynamically unloaded.
- Fixed a hang if the network plugin returned an error.
- Fixed a hang on heterogeneous architectures by harmonizing tuning choices.
- Fixed a potential crash in case of a failed communicator initialization or termination.
- Fixed a potential bug during a group launch of multiple communicators.
- Fixed a bug where, under rare circumstances, certain variables specified in the config file could be ignored.

Updating the GPG Repository Key

To best ensure the security and reliability of our RPM and Debian package repositories, NVIDIA is updating and rotating the signing keys used by apt, dnf/yum, and zypper

package managers beginning on April 27, 2022. Failure to update your repository signing keys will result in package management errors when attempting to access or install NCCL packages. To ensure continued access to the latest NCCL release, please follow the updated NCCL installation guide.

Chapter 20. NCCL Release 2.25.1

This is the NCCL 2.25.1 release notes. For previous NCCL release notes, refer to the NCCL Archives.

Compatibility

NCCL 2.25.1 has been tested with the following:

- Deep learning framework containers. Refer to the Support Matrix for the supported container version.
- This NCCL release supports CUDA 12.2, CUDA 12.4, and CUDA 12.8.

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- Added support for Blackwell.
- Improved NVTX instrumentation.

Updating the GPG Repository Key

To best ensure the security and reliability of our RPM and Debian package repositories, NVIDIA is updating and rotating the signing keys used by apt, dnf/yum, and zypper package managers beginning on April 27, 2022. Failure to update your repository signing keys will result in package management errors when attempting to access or install NCCL packages. To ensure continued access to the latest NCCL release, please follow the updated NCCL installation guide.

Chapter 21. NCCL Release 2.24.3

This is the NCCL 2.24.3 release notes. For previous NCCL release notes, refer to the NCCL Archives.

Compatibility

NCCL 2.24.3 has been tested with the following:

- Deep learning framework containers. Refer to the Support Matrix for the supported container version.
- This NCCL release supports CUDA 12.2, CUDA 12.4, and CUDA 12.6.

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- Leverage user buffer registration for inter-node copy operations using the ring or tree algorithm.
- Added RAS subsystem and ncclras tool to allow for a report of the NCCL state in case of a hang.
- Added support for 8bit floating point data types (e5m2 and e4m3).
- Added NIC fusion: fuse multiple NICs together as a single, larger one.
- Retry in case of socket connection failure (unreachable host).
- Retry in case of IB QP connection failure.
- Improved support for external network plugins: allow plugins to force a flush, indicate when completion is not needed, allow for full offload of allgather operations when using one GPU per node.
- Extended `NCCL_ALGO/NCCL_PROTO` syntax and strictly enforce their values.
- Made host memory allocation use cumem functions by default.

Fixed Issues

The following issues have been resolved in NCCL 2.24.3:

- Return `ncclInvalidUsage` when `NCCL_SOCKET_IFNAME` is set to an incorrect value.

- Fixed PAT tuning.
- Fixed hangs when running with different CPU architectures.
- Fixed FD leak in UDS.
- Fixed crash when mixing buffer registration and graph buffer registration.
- Made ncclSend/ncclRecv communication with buffer registration functional on network plugins relying on dmabuf for buffer registration.
- Fixed crash in IB code caused by uninitialized fields.
- Fixed case where `ncclSend/ncclRecv` would return `ncclSuccess` in non-blocking mode even though the operation was not enqueued onto the stream (Github issue 1495).

Updating the GPG Repository Key

To best ensure the security and reliability of our RPM and Debian package repositories, NVIDIA is updating and rotating the signing keys used by apt, dnf/yum, and zypper package managers beginning on April 27, 2022. Failure to update your repository signing keys will result in package management errors when attempting to access or install NCCL packages. To ensure continued access to the latest NCCL release, please follow the updated NCCL installation guide.

Chapter 22. NCCL Release 2.23.4

This is the NCCL 2.23.4 release notes. For previous NCCL release notes, refer to the NCCL Archives.

Compatibility

NCCL 2.23.4 has been tested with the following:

- Deep learning framework containers. Refer to the Support Matrix for the supported container version.
- This NCCL release supports CUDA 12.2, CUDA 12.4, and CUDA 12.6.

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- Add a new scalable initialization API.
- Accelerate bootstrap operations.
- Add new PAT (Parallel Aggregated Trees) algorithm for allgather and reduce scatter operations using one GPU per node.
- Accelerate intra-node communication when buffers are registered.
- Add profiler plugin API.
- Make CUDA calls within graph allocation asynchronous.
- Use fatal RDMA asynchronous events to stop network operations early.
- Disable GPU Direct P2P on AMD CPUs when using more than 2 GPUs.
- Add parameter to set the location of the user configuration file.
- Increase default IB timeout from 18 to 20.

Fixed Issues

The following issues have been resolved in NCCL 2.23.4:

- Fixed GPU Direct RDMA check on linux kernels 6.6+.
- Fixed performance regression when mixing small and large operations.
- Fixed crash in topology detection when devices have a NUMA ID of -1.

- Fixed Tree graph search when NCCL_CROSS_NIC is set to 1..
- Fixes for IB operation on multi-node NVLink systems.

Updating the GPG Repository Key

To best ensure the security and reliability of our RPM and Debian package repositories, NVIDIA is updating and rotating the signing keys used by apt, dnf/yum, and zypper package managers beginning on April 27, 2022. Failure to update your repository signing keys will result in package management errors when attempting to access or install NCCL packages. To ensure continued access to the latest NCCL release, please follow the updated NCCL installation guide.

Chapter 23. NCCL Release 2.22.3

This is the NCCL 2.22.3 release notes. For previous NCCL release notes, refer to the NCCL Archives.

Compatibility

NCCL 2.22.3 has been tested with the following:

- Deep learning framework containers. Refer to the Support Matrix for the supported container version.
- This NCCL release supports CUDA 12.2, CUDA 12.4, and CUDA 12.5.

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- Lazy connection establishment for collective operations to speed up init and lower memory usage when not using all algorithms.
- Accelerated Intra-node NVLink detection for faster init.
- Init profiling, reporting time spent in different init phases.
- Support for PCI p2p on split PCI switches.
- Cost estimation API.
- Net/IB: separate traffic class for fifo messages.
- Net/IB: support for IB router.
- Optimizations and fixes for device network offload (unpack).
- Support ncclGroupStart/End for ncclCommAbort/Destroy.
- Improved Tuner API.
- Allow net and tuner plugins to be statically linked.

Fixed Issues

The following issues have been resolved in NCCL 2.22.3:

- Disabled port fusion in heterogeneous systems.
- Fixed NVLS graphs search for dual NICs.

- Fixed crash with collnetDirect.
- Fixed hang when nodes have different CPU types.
- Fixed performance of registered send/recv operations.
- Ensured ncclCommDestroy is non-blocking if ncclCommFinalize has been called.
- Improved error reporting during SHM segment creation.
- Improved support of various compilers.

Updating the GPG Repository Key

To best ensure the security and reliability of our RPM and Debian package repositories, NVIDIA is updating and rotating the signing keys used by apt, dnf/yum, and zypper package managers beginning on April 27, 2022. Failure to update your repository signing keys will result in package management errors when attempting to access or install NCCL packages. To ensure continued access to the latest NCCL release, please follow the updated NCCL installation guide.

Chapter 24. NCCL Release 2.21.5

This is the NCCL 2.21.5 release notes. For previous NCCL release notes, refer to the NCCL Archives.

Compatibility

NCCL 2.21.5 has been tested with the following:

- Deep learning framework containers. Refer to the Support Matrix for the supported container version.
- This NCCL release supports CUDA 11.0, CUDA 12.2, CUDA 12.4, and CUDA 12.5.

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- Added support for user buffer registrations on IB SHARP operations with one GPU per node.
- Improved support for Multi-node NVLink systems, add NVLink SHARP support and multi-click support.
- Added support for dynamic GID detection on RoCE.
- Reduced memory usage when NVLink SHARP is enabled.
- Improved tuner plugin loading.
- Added signature to communicator objects to help detect the corruption of communicator objects or invalid communicator pointers.

Fixed Issues

The following issues have been resolved in NCCL 2.21.5:

- Fixed IB SHARP rail mapping when using split communicators.
- Fixed mismatch crash during bootstrap due to TCP packet reordering.
- Fixed hang with heterogeneous systems, causing the crossNic value to be different between nodes.
- Fixed minCompCap/maxCompCap computation.

Updating the GPG Repository Key

To best ensure the security and reliability of our RPM and Debian package repositories, NVIDIA is updating and rotating the signing keys used by apt, dnf/yum, and zypper package managers beginning on April 27, 2022. Failure to update your repository signing keys will result in package management errors when attempting to access or install NCCL packages. To ensure continued access to the latest NCCL release, please follow the updated NCCL installation guide.

Chapter 25. NCCL Release 2.20.5

This is the NCCL 2.20.5 release notes. For previous NCCL release notes, refer to the NCCL Archives.

Compatibility

NCCL 2.20.5 has been tested with the following:

- Deep learning framework containers. Refer to the Support Matrix for the supported container version.
- This NCCL release supports CUDA 11.0, CUDA 12.2, and CUDA 12.4.

Fixed Issues

The following issues have been resolved in NCCL 2.20.5:

- Fixed crash in UDS connection when using `ncclCommSplit`.

Updating the GPG Repository Key

To best ensure the security and reliability of our RPM and Debian package repositories, NVIDIA is updating and rotating the signing keys used by apt, dnf/yum, and zypper package managers beginning on April 27, 2022. Failure to update your repository signing keys will result in package management errors when attempting to access or install NCCL packages. To ensure continued access to the latest NCCL release, please follow the updated NCCL installation guide.

Chapter 26. NCCL Release 2.20.3

This is the NCCL 2.20.3 release notes. For previous NCCL release notes, refer to the NCCL Archives.

Compatibility

NCCL 2.20.3 has been tested with the following:

- Deep learning framework containers. Refer to the Support Matrix for the supported container version.
- This NCCL release supports CUDA 11.0, CUDA 12.2, and CUDA 12.3.

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- Improved ring algorithm (alternating rings), to remove bottleneck on systems where each GPU only has one local NIC, like the DGX H100.
- Added support for user buffer registration for network send/recv operations.
- Optimized aggregated operations to better utilize all channels.
- Added support for large Broadcom PCI gen5 switches, flattening the reported two-level topology.
- Added support for inter-node NVLink communication.
- Add support for port fusion in NET/IB.
- Added support for ReduceScatter and AllGather using IB SHARP.

Fixed Issues

The following issues have been resolved in NCCL 2.20.3:

- Fixed hang during A2A connection.

Updating the GPG Repository Key

To best ensure the security and reliability of our RPM and Debian package repositories, NVIDIA is updating and rotating the signing keys used by apt, dnf/yum, and zypper package managers beginning on April 27, 2022. Failure to update your repository signing

keys will result in package management errors when attempting to access or install NCCL packages. To ensure continued access to the latest NCCL release, please follow the updated NCCL installation guide.

Chapter 27. NCCL Release 2.19.3

This is the NCCL 2.19.3 release notes. For previous NCCL release notes, refer to the NCCL Archives.

Compatibility

NCCL 2.19.3 has been tested with the following:

- Deep learning framework containers. Refer to the Support Matrix for the supported container version.
- This NCCL release supports CUDA 11.0, CUDA 12.0, CUDA 12.2, and CUDA 12.3.

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- Added local user buffer registration for NVLink SHARP.
- Improved performance on Hopper GPUs (H800/H100).
- Added tuning plugin support.
- Added NET API v7 to allow for device side packet reordering; removed v4 plugin support.
- Added support for RoCE ECE.
- Added support for C2C links.
- Disabled network flush by default on H100.

Fixed Issues

The following issues have been resolved in NCCL 2.19.3:

- Better detection of shared memory allocation failures and report them instead of crashing later with a "Bus error."
- Fixed missing thread unlock in bootstrap code.

Known Issues

- Alltoall performance at scale may see degradation for medium-size operations. Setting `NCCL_NCHANNELS_PER_NET_PEER=1` should workaround the problem.
- A hang may occur during alltoall connection establishment between multiple nodes (more likely at scale). This hang can be avoided by setting `NCCL_CUMEM_ENABLE=0`.

Updating the GPG Repository Key

To best ensure the security and reliability of our RPM and Debian package repositories, NVIDIA is updating and rotating the signing keys used by apt, dnf/yum, and zypper package managers beginning on April 27, 2022. Failure to update your repository signing keys will result in package management errors when attempting to access or install NCCL packages. To ensure continued access to the latest NCCL release, please follow the updated NCCL installation guide.

Chapter 28. NCCL Release 2.18.5

This is the NCCL 2.18.5 release notes. For previous NCCL release notes, refer to the NCCL Archives.

Compatibility

NCCL 2.18.5 has been tested with the following:

- Deep learning framework containers. Refer to the Support Matrix for the supported container version.
- This NCCL release supports CUDA 11.0, CUDA 12.0, and CUDA 12.2.

Fixed Issues

The following issues have been resolved in NCCL 2.18.5:

- Fixed NVLS search issues.
- Increased Max IB network interfaces to 32.
- Fixed inconsistent network device ordering when creating communicators with only one GPU per node.
- Try to have different GPUs use all network interfaces on systems with more than one network interface per GPU.

Known Issues

- Send/receive communication using `CUDA_VISIBLE_DEVICES` and PXN only works if the GPU mappings to local ranks is the same across nodes. Disabing PXN for Send/Receive communication can workaround the issue (`NCCL_P2P_PXN_LEVEL=0`).

Updating the GPG Repository Key

To best ensure the security and reliability of our RPM and Debian package repositories, NVIDIA is updating and rotating the signing keys used by apt, dnf/yum, and zypper package managers beginning on April 27, 2022. Failure to update your repository signing keys will result in package management errors when attempting to access or install NCCL packages. To ensure continued access to the latest NCCL release, please follow the updated NCCL installation guide.

Chapter 29. NCCL Release 2.18.3

This is the NCCL 2.18.3 release notes. For previous NCCL release notes, refer to the NCCL Archives.

Compatibility

NCCL 2.18.3 has been tested with the following:

- Deep learning framework containers. Refer to the Support Matrix for the supported container version.
- This NCCL release supports CUDA 11.0, CUDA 12.0, CUDA 12.1, and CUDA 12.2.

Fixed Issues

The following issues have been resolved in NCCL 2.18.3:

- Fixed data corruption on DGX/HGX H100 systems when using LL128 protocol.
- Fixed hang with IB SHARP and bfloat16 on systems with less than one NIC per GPU.
- Fixed regression in initialization time.
- Fixed data corruption with IB SHARP on H100 platforms when combining multiple GPUs per process and multiple processes per node.
- Fixed crash when shared memory creation fails.
- Fixed Avg operation with IB SHARP when using Collnet/Chain algorithm.
- Fixed performance for all-to-all operations at large scale on systems with more than one NIC per GPU.
- Fixed performance on DGX H800.
- Fixed race condition in connection progress that caused a crash.
- Fixed network flush with IB SHARP.
- Fixed PXN operation when `CUDA_VISIBLE_DEVICES` is set.
- Fixed performance of aggregated reduceScatter/allGather operations.

Known Issues

- Send/receive communication using `CUDA_VISIBLE_DEVICES` and PXN only works if the GPU mappings to local ranks is the same across nodes. Disabing PXN for Send/Receive communication can workaround the issue (`NCCL_P2P_PXN_LEVEL=0`).

Updating the GPG Repository Key

To best ensure the security and reliability of our RPM and Debian package repositories, NVIDIA is updating and rotating the signing keys used by apt, dnf/yum, and zypper package managers beginning on April 27, 2022. Failure to update your repository signing keys will result in package management errors when attempting to access or install NCCL packages. To ensure continued access to the latest NCCL release, please follow the updated NCCL installation guide.

Chapter 30. NCCL Release 2.18.1

This is the NCCL 2.18.1 release notes. For previous NCCL release notes, refer to the NCCL Archives.

Compatibility

NCCL 2.18.1 has been tested with the following:

- Deep learning framework containers. Refer to the Support Matrix for the supported container version.
- This NCCL release supports CUDA 11.0, CUDA 12.0, and CUDA 12.1.

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- Add inter-node algorithms for NVLink SHARP: NVLink SHARP + IB SHARP (NVLS), NVLink SHARP + Tree (NVLSTREE).
- Add `ncclCommSplit` primitive, with optional resource sharing.
- Add support for memory management using cuMem functions (disabled by default).
- Add option to use multiple QPs in round-robin mode.

Known Issues

- On systems where a NIC shares a PCI switch with only one GPU (like on HGX H100), the Tree algorithm will make data transit through the CPU, making the LL128 protocol unsafe. This could result in data corruption. You can workaround this issue by setting the following:

  ```
  NCCL_IB_PCI_RELAXED_ORDERING=0
  ```

  Another solution is to disable the LL128 protocol with the following:

  ```
  NCCL_PROTO=^LL128
  ```
- On systems with less than 1 NIC per GPU, running `bfloat16` reductions with IB SHARP will cause a hang.
- Running allreduce on H100 platforms with IB SHARP with multiple GPUs per process can result in data corruption if all GPUs on the node are not part of the same process, or cannot access other GPUs buffers directly.

Fixed Issues

The following issues have been resolved in NCCL 2.18.1:

- Fixed hangs with irregular send/receive patterns (e.g., `alltoallv`).
- Use all NICs for Send/Receive operations on systems with more than one NIC per GPU.
- Increased number of channels on H100 for network communication when bandwidth is not limited by NVLink bandwidth.
- Improved error reporting in case of IB Verbs errors.
- Fixed context creation for progress thread.
- Fixed hang in `commReclaim`.
- Fixed performance issue when NVB was disabled.

Updating the GPG Repository Key

To best ensure the security and reliability of our RPM and Debian package repositories, NVIDIA is updating and rotating the signing keys used by apt, dnf/yum, and zypper package managers beginning on April 27, 2022. Failure to update your repository signing keys will result in package management errors when attempting to access or install NCCL packages. To ensure continued access to the latest NCCL release, please follow the updated NCCL installation guide.

Chapter 31. NCCL Release 2.17.1

This is the NCCL 2.17.1 release notes. For previous NCCL release notes, refer to the NCCL Archives.

Compatibility

NCCL 2.17.1 has been tested with the following:

- Deep learning framework containers. Refer to the Support Matrix for the supported container version.
- This NCCL release supports CUDA 11.0, CUDA 12.0, and CUDA 12.1.

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- Add support for NVLink SHARP Reduction / Broadcast to accelerate intra-node allreduce operations.
- Add new fields in the communicator configuration structure: Cooperative Group Array cluster size, minimum and maximum number of CTAs, network plugin name to use.
- Update NVTX3 includes.

Known Issues

- On systems where a NIC shares a PCI switch with only one GPU (like on HGX H100), the Tree algorithm will make data transit through the CPU, making the LL128 protocol unsafe. This could result in data corruption. You can workaround this issue by setting the following:

  ```
  NCCL_IB_PCI_RELAXED_ORDERING=0
  ```

  Another solution is to disable the LL128 protocol with the following:

  ```
  NCCL_PROTO=^LL128
  ```

- Performance is sub-optimal for NCCL communicators using 1 NIC per node on DGX H100/HGX H100 + NDR. It can be worked around by setting the following parameter:

  ```
  NCCL_MIN_NCHANNELS=4
  ```

Fixed Issues

The following issues have been resolved in NCCL 2.17.1:

- Fix crash when one CollNet (SHARP) rail fails to initialize.
- Re-enable the LL128 protocol on H100 when we use PXN to close rings.

Updating the GPG Repository Key

To best ensure the security and reliability of our RPM and Debian package repositories, NVIDIA is updating and rotating the signing keys used by apt, dnf/yum, and zypper package managers beginning on April 27, 2022. Failure to update your repository signing keys will result in package management errors when attempting to access or install NCCL packages. To ensure continued access to the latest NCCL release, please follow the updated NCCL installation guide.

Chapter 32. NCCL Release 2.16.5

This is the NCCL 2.16.5 release notes. For previous NCCL release notes, refer to the NCCL Archives.

Compatibility

NCCL 2.16.5 has been tested with the following:

- Deep learning framework containers. Refer to the Support Matrix for the supported container version.
- This NCCL release supports CUDA 11.0, CUDA 11.8, and CUDA 12.0..

Known Issues

- Performance is sub-optimal for NCCL communicators using 1 NIC per node on DGX H100/HGX H100 + NDR. It can be worked around by setting the following parameter:
  ```
  NCCL_MIN_NCHANNELS=4
  ```

Fixed Issues

The following issues have been resolved in NCCL 2.16.5:

- Fix speed of IB NDR links
- Fix handling of EINTR in socket polling
- Improve proxy progress scheduling
- Fix resource cleanup
- Fix double free on case of init failure
- Fix crash in case of communicator abort
- Revert performance optimization causing significant regressions on some AMD platforms

Updating the GPG Repository Key

To best ensure the security and reliability of our RPM and Debian package repositories, NVIDIA is updating and rotating the signing keys used by apt, dnf/yum, and zypper package managers beginning on April 27, 2022. Failure to update your repository signing

keys will result in package management errors when attempting to access or install NCCL packages. To ensure continued access to the latest NCCL release, please follow the updated NCCL installation guide.

Chapter 33. NCCL Release 2.16.2

This is the NCCL 2.16.2 release notes. For previous NCCL release notes, refer to the NCCL Archives.

Compatibility

NCCL 2.16.2 has been tested with the following:

- Deep learning framework containers. Refer to the Support Matrix for the supported container version.
- This NCCL release supports CUDA 11.0, CUDA 11.8, and CUDA 12.0..

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- Add support for CUDA 12.0
- Make socket support more resistant to network scanners
- Improve performance on large CUDA graphs, reducing dependencies
- Compile with profiling API by default
- Extend NVTX instrumentation with call arguments

Fixed Issues

The following issues have been resolved in NCCL 2.16.2:

- Various fixes to `ncclCommAbort`
- Make service thread polling resistant to EINTR
- Adjust inter-socket AMD bandwidth model to favor faster paths

Updating the GPG Repository Key

To best ensure the security and reliability of our RPM and Debian package repositories, NVIDIA is updating and rotating the signing keys used by apt, dnf/yum, and zypper package managers beginning on April 27, 2022. Failure to update your repository signing keys will result in package management errors when attempting to access or install

NCCL packages. To ensure continued access to the latest NCCL release, please follow the updated NCCL installation guide.

Chapter 34. NCCL Release 2.15.5

This is the NCCL 2.15.5 release notes. For previous NCCL release notes, refer to the NCCL Archives.

Compatibility

NCCL 2.15.5 has been tested with the following:

- Deep learning framework containers. Refer to the Support Matrix for the supported container version.
- This NCCL release supports CUDA 10.2, CUDA 11.0, and CUDA 11.8.

Fixed Issues

The following issues have been resolved in NCCL 2.15.5:

- Fix crash with CollnetChain on some node topologies
- Fix hang when interleaving the capture of different graphs
- Fix hang during init in multi-threaded mode
- Fix potential data corruption with LL128 protocol on unaligned buffers
- Fix CPU usage during preconnect
- Fixes double-free in the error path for ncclCommInitAll
- Workaround hang on H100 with Ring/LL128 on 2 GPUs

Updating the GPG Repository Key

To best ensure the security and reliability of our RPM and Debian package repositories, NVIDIA is updating and rotating the signing keys used by apt, dnf/yum, and zypper package managers beginning on April 27, 2022. Failure to update your repository signing keys will result in package management errors when attempting to access or install NCCL packages. To ensure continued access to the latest NCCL release, please follow the updated NCCL installation guide.

Chapter 35. NCCL Release 2.15.1

This is the NCCL 2.15.1 release notes. For previous NCCL release notes, refer to the NCCL Archives.

Compatibility

NCCL 2.15.1 has been tested with the following:

- Deep learning framework containers. Refer to the Support Matrix for the supported container version.
- This NCCL release supports CUDA 10.2, CUDA 11.0, and CUDA 11.8.

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- Support for H100 (sm90).

Fixed Issues

The following issues have been resolved in NCCL 2.15.1:

- Make sure NCCL kernel honors user stream priorities.

Updating the GPG Repository Key

To best ensure the security and reliability of our RPM and Debian package repositories, NVIDIA is updating and rotating the signing keys used by apt, dnf/yum, and zypper package managers beginning on April 27, 2022. Failure to update your repository signing keys will result in package management errors when attempting to access or install NCCL packages. To ensure continued access to the latest NCCL release, please follow the updated NCCL installation guide.

Chapter 36. NCCL Release 2.14.3

This is the NCCL 2.14.3 release notes. For previous NCCL release notes, refer to the NCCL Archives.

Compatibility

NCCL 2.14.3 has been tested with the following:

- Deep learning framework containers. Refer to the Support Matrix for the supported container version.
- This NCCL release supports CUDA 10.2, CUDA 11.0, and CUDA 11.7.

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- Add support for improved fault tolerance: non-blocking mode, new init function with configuration, new finalize function.
- Reintroduce the collnet+chain algorithm.
- Add LL protocol for intra-node send/recv communication, and inter-node (disabled by default).
- Communicate through the network within a node instead of shared memory if performance would be better that way.

Fixed Issues

The following issues have been resolved in NCCL 2.14.3:

- Wait for CUDA graph destruction before freeing communicator.
- Remove aggressive polling during enqueue.
- Fix DMABUF fallback on MOFED 5.4 and earlier.
- Fix `NCCL_DEBUG_FILE` functionality.

Updating the GPG Repository Key

To best ensure the security and reliability of our RPM and Debian package repositories, NVIDIA is updating and rotating the signing keys used by apt, dnf/yum, and zypper

package managers beginning on April 27, 2022. Failure to update your repository signing keys will result in package management errors when attempting to access or install NCCL packages. To ensure continued access to the latest NCCL release, please follow the updated NCCL installation guide.

Chapter 37. NCCL Release 2.13.4

This is the NCCL 2.13.4 release notes. For previous NCCL release notes, refer to the NCCL Archives.

Compatibility

NCCL 2.13.4 has been tested with the following:

- Deep learning framework containers. Refer to the Support Matrix for the supported container version.
- This NCCL release supports CUDA 10.2, CUDA 11.0, and CUDA 11.7.

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- Optimize CUDA graph launch; avoid launching a CPU callback for intra-node operations.
- Simplify kernel common code to improve the latency of send/recv operations.
- Strengthen CUDA streams semantics.
- Change NET API to v6, to add dmabuf support.
- Add `ncclGetLastError()` function.
- Add `ncclRemoteError` code and use it for remote network errors.
- Support the use of a different `NCCL_NET` parameter per communicator.
- Add support for SHM and P2P transfers using cudaMemcpy.

Fixed Issues

The following issues have been resolved in NCCL 2.13.4:

- Fix multi-receive size encoding which could cause flush to be skipped in corner cases mixing zero-bytes send/receive operations and non-zero-bytes send/receive operations.
- Replace busy polling in the bootstrap thread waiting for ranks to check in by a blocking accept.

Updating the GPG Repository Key

To best ensure the security and reliability of our RPM and Debian package repositories, NVIDIA is updating and rotating the signing keys used by apt, dnf/yum, and zypper package managers beginning on April 27, 2022. Failure to update your repository signing keys will result in package management errors when attempting to access or install NCCL packages. To ensure continued access to the latest NCCL release, please follow the updated NCCL installation guide.

Chapter 38. NCCL Release 2.12.12

This is the NCCL 2.12.12 release notes. For previous NCCL release notes, refer to the NCCL Archives.

Compatibility

NCCL 2.12.12 has been tested with the following:

- Deep learning framework containers. Refer to the Support Matrix for the supported container version.
- This NCCL release supports CUDA 10.2, CUDA 11.0, CUDA 11.6, and CUDA 11.7.

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- Improved allreduce performance when you have more than one network interface per GPU and you need to use PXN to close rings.
- Added `P2P_DIRECT_DISABLE` parameter to disable direct access to pointers within a process.
- Added support for PCI Gen5 on 5.4 kernels.

Fixed Issues

The following issues have been resolved in NCCL 2.12.12:

- Fixed hang on cubemesh topologies.
- Fixed random crash in `init` due to uninitialized `struct`.
- Fixed crash when setting `NCCL_SET_THREAD_NAME`.

Updating the GPG Repository Key

To best ensure the security and reliability of our RPM and Debian package repositories, NVIDIA is updating and rotating the signing keys used by apt, dnf/yum, and zypper package managers beginning on April 27, 2022. Failure to update your repository signing keys will result in package management errors when attempting to access or install

NCCL packages. To ensure continued access to the latest NCCL release, please follow the updated NCCL installation guide.

Chapter 39. NCCL Release 2.12.10

This is the NCCL 2.12.10 release notes. For previous NCCL release notes, refer to the NCCL Archives.

Compatibility

NCCL 2.12.10 has been tested with the following:

- Deep learning framework containers. Refer to the Support Matrix for the supported container version.
- This NCCL release supports CUDA 10.2, CUDA 11.0, and CUDA 11.6.

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- Improved error messages for network errors, with hostname instead of IP address.

Fixed Issues

The following issues have been resolved in NCCL 2.12.10:

- Fixed bug with collNet.
- Fixed bug with zero-byte send/recv operations.
- Fixed `NCCL_PARAM` implementation taking a lock every time a parameter value was queried.
- Fixed bug when setting `NCCL_IB_QPS_PER_CONNECTION` to more than one.

Chapter 40. NCCL Release 2.12.7

This is the NCCL 2.12.7 release notes. For previous NCCL release notes, refer to the NCCL Archives.

Compatibility

NCCL 2.12.7 has been tested with the following:

- Deep learning framework containers. Refer to the Support Matrix for the supported container version.
- This NCCL release supports CUDA 10.2, CUDA 11.0, and CUDA 11.6.

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- Added NVLink-optimized network communication to keep traffic rail-local (PXN).
- Improved alltoall latency by aggregating messages within a node to a given destination.
- Added new v5 plugin API with grouped receives and tags, keeping compatibility for v4 plugins.
- Added naming of NCCL threads to help debugging.
- Added support for Relaxed Ordering for IB.
- Added profiling and timing infrastructure.

Fixed Issues

The following issues have been resolved in NCCL 2.12.7:

- Fixed NVLink detection and avoid data corruption when some NVLinks are down.

Chapter 41. NCCL Release 2.11.4

This is the NCCL 2.11.4 release notes. For previous NCCL release notes, refer to the NCCL Archives.

Compatibility

NCCL 2.11.4 has been tested with the following:

- Deep learning framework containers. Refer to the Support Matrix for the supported container version.
- This NCCL release supports CUDA 10.2, CUDA 11.0, CUDA 11.4, CUDA 11.5, and CUDA 11.6.

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- Added new API for creating a reduction operation which multiplies the input by a rank-specific scalar before doing an inter-rank summation (see: `ncclRedOpCreatePreMulSum`).
- Improved CollNet (SHARP) performance of `ncclAllReduce` when captured in a CUDA Graph via user buffer registration.
- Added env `NCCL_NET_PLUGIN="<suffix>"` to allow the user a way to choose among multiple NCCL net plugins by substituting into `libnccl-net-<suffix>.so`.

Fixed Issues

The following issues have been resolved in NCCL 2.11.4:

- Fixed memory leak of NVB connections.
- Fixed crash of `ncclGroup()` containing mixed datatypes/operations (GitHub issue #560, introduced in NCCL 2.10.3).
- Fixed topology detection of IB Virtual Functions (SR-IOV).

Chapter 42. NCCL Release 2.10.3

This is the NCCL 2.10.3 release notes. For previous NCCL release notes, refer to the NCCL Archives.

Compatibility

NCCL 2.10.3 has been tested with the following:

- Deep learning framework containers. Refer to the Support Matrix for the supported container version.
- This NCCL release supports CUDA 10.2, CUDA 11.0, and CUDA 11.4.

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- Added ncclAvg operation
- Added support for bfloat16 type
- Added support for multiple IB queue pairs
- Added WSL2 support (single GPU only)
- Improved performance for aggregation
- Improved performance for medium sizes
- Improved network error reporting
- Added NCCL_NET environment variable to use a specific network
- Added auto-load of XML topology from default location

Fixed Issues

The following issues have been resolved in NCCL 2.10.3:

- Fixed graph search on cubemesh topologies.
- Fixed all-to-all affinity to improve latency.
- Fixed hang in cubemesh NVB connections during initialization.

Chapter 43. NCCL Release 2.9.9

This is the NCCL 2.9.9 release notes. For previous NCCL release notes, refer to the NCCL Archives.

Compatibility

NCCL 2.9.9 has been tested with the following:

- Deep learning framework containers. Refer to the Support Matrix for the supported container version.
- This NCCL release supports CUDA 10.1, CUDA 10.2, CUDA 11.0, and CUDA 11.3.

Fixed Issues

The following issues have been resolved in NCCL 2.9.9:

- Fixed crash when setting `NCCL_MAX_P2P_NCHANNELS`.
- Fixed hang during sendrecv dynamic connection on cubemesh topologies.
- Fixed compilation with `TRACE=1`.

Chapter 44. NCCL Release 2.9.8

This is the NCCL 2.9.8 release notes. For previous NCCL release notes, refer to the NCCL Archives.

Compatibility

NCCL 2.9.8 has been tested with the following:

- Deep learning framework containers. Refer to the Support Matrix for the supported container version.
- This NCCL release supports CUDA 10.1, CUDA 10.2, CUDA 11.0, and CUDA 11.3.

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- Added support for `nvidia-peermem` module.

Fixed Issues

The following issues have been resolved in NCCL 2.9.8:

- Fixed error when injecting a topology file without vendor/device information.
- Fixed crash in bootstrap error case.
- Fixed memory leaks.
- Fixed `collnet` clean-up issue.

Chapter 45. NCCL Release 2.9.6

This is the NCCL 2.9.6 release notes. For previous NCCL release notes, refer to the NCCL Archives.

Compatibility

NCCL 2.9.6 has been tested with the following:

- Deep learning framework containers. Refer to the Support Matrix for the supported container version.
- This NCCL release supports CUDA 10.1, CUDA 10.2, CUDA 11.0, and CUDA 11.3.

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- Added support for CUDA graphs
- Improved performance for CollNet (SHARP)
- Fuse PCI Gen4 switches showing a two-level hierarchy into a single level.
- Improve NIC balancing for communicators using a single GPU per node.

Fixed Issues

The following issues have been resolved in NCCL 2.9.6:

- Fixed bootstrap hang in case of reordered packets causing connections to be inverted.
- Fix locking issue causing NCCL calls to block until previous operations were complete.

Chapter 46. NCCL Release 2.8.4

This is the NCCL 2.8.4 release notes. For previous NCCL release notes, refer to the NCCL Archives.

Compatibility

NCCL 2.8.4 has been tested with the following:

- Deep learning framework containers. Refer to the Support Matrix for the supported container version.
- This NCCL release supports CUDA 10.1, CUDA 10.2, CUDA 11.0, CUDA 11.1, and CUDA 11.2.

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- Added support for Zhaoxin CPUs

Known Issues

Send/receive operations have a number of limitations:

- Using send/receive operations in combination to launch work on multiple GPUs from a single process can fail or hang if the GPUs process different amounts of data. Setting `NCCL_LAUNCH_MODE=PARALLEL` can work around the issue, but can also cause other problems. For more information, see the NCCL User Guide section Troubleshooting > Known Issues > Concurrency Between NCCL and CUDA calls.

Fixed Issues

The following issues have been resolved in NCCL 2.8.4:

- Fixed hang for some imbalanced send/recv operation (alltoallv).

Chapter 47. NCCL Release 2.8.3

This is the NCCL 2.8.3 release notes. For previous NCCL release notes, refer to the NCCL Archives.

Compatibility

NCCL 2.8.3 has been tested with the following:

- Deep learning framework containers. Refer to the Support Matrix for the supported container version.
- This NCCL release supports CUDA 10.1, CUDA 10.2, CUDA 11.0, CUDA 11.1, and CUDA 11.2.

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- Optimized Tree performance on A100
- Improved performance for aggregated operations
- Improved performance for all-to-all operations at scale
- Reduced memory usage for all-to-all operations at scale
- Optimized all-to-all performance on DGX-1

Known Issues

Send/receive operations have a number of limitations:

- Using send/receive operations in combination to launch work on multiple GPUs from a single process can fail or hang if the GPUs process different amounts of data. Setting `NCCL_LAUNCH_MODE=PARALLEL` can work around the issue, but can also cause other problems. For more information, see the NCCL User Guide section Troubleshooting > Known Issues > Concurrency Between NCCL and CUDA calls.

Fixed Issues

The following issues have been resolved in NCCL 2.8.3:

- Hang in LL128 protocol after 2^31 steps.

- Topology injection error when using fewer GPUs than described. (github issue #379)
- Protocol mismatch causing hangs or crashes when using one GPU per node. (github issue #394)

Chapter 48. NCCL Release 2.7.8

This is the NCCL 2.7.8 release notes. For previous NCCL release notes, refer to the NCCL Archives.

Compatibility

NCCL 2.7.8 has been tested with the following:

- Deep learning framework containers. Refer to the Support Matrix for the supported container version.
- This NCCL release supports CUDA 10.1, CUDA 10.2, CUDA 11.0, amd CUDA 11.1.

Known Issues

Send/receive operations have a number of limitations:

- Using send/receive operations in combination to launch work on multiple GPUs from a single process can fail or hang if the GPUs process different amounts of data. Setting `NCCL_LAUNCH_MODE=PARALLEL` can work around the issue, but can also cause other problems. For more information, see the NCCL User Guide section Troubleshooting > Known Issues > Concurrency Between NCCL and CUDA calls.
- Aggregation is not supported on a given source-destination pair, meaning that there can only be one send/receive per source-destination pair.
- Each source-destination pair allocates a dedicated fifo of 4 MB (see `NCCL_BUFFSIZE` variable) which can use a significant amount of GPU memory at scale.
- When using GPU Direct RDMA, each point-to-point connection will also use resources on the GPU PCI address space, which we might run out of on some models. If that happens, consider disabling GPU Direct RDMA (`NCCL_NET_GDR_LEVEL=0`) or reduce the per-peer buffer size (`NCCL_BUFFSIZE`).
- Send/receive operations are not yet optimized for the DGX-1 NVLink topology.

Fixed Issues

The following issues have been resolved in NCCL 2.7.8:

- Resolved "Collective Mismatch" errors reported erroneously when using send/recv operations.

Chapter 49. NCCL Release 2.7.6

This is the NCCL 2.7.6 release notes. For previous NCCL release notes, refer to the NCCL Archives.

Compatibility

NCCL 2.7.6 has been tested with the following:

- Deep learning framework containers. Refer to the Support Matrix for the supported container version.
- This NCCL release supports CUDA 10.1, CUDA 10.2, and CUDA 11.0.

Known Issues

Send/receive operations have a number of limitations:

- Using send/receive operations in combination to launch work on multiple GPUs from a single process can fail or hang if the GPUs process different amounts of data. Setting `NCCL_LAUNCH_MODE=PARALLEL` can work around the issue, but can also cause other problems. For more information, see the NCCL User Guide section Troubleshooting > Known Issues > Concurrency Between NCCL and CUDA calls.
- Aggregation is not supported on a given source-destination pair, meaning that there can only be one send/receive per source-destination pair.
- Each source-destination pair allocates a dedicated fifo of 4 MB (see `NCCL_BUFFSIZE` variable) which can use a significant amount of GPU memory at scale.
- When using GPU Direct RDMA, each point-to-point connection will also use resources on the GPU PCI address space, which we might run out of on some models. If that happens, consider disabling GPU Direct RDMA (`NCCL_NET_GDR_LEVEL=0`) or reduce the per-peer buffer size (`NCCL_BUFFSIZE`).
- Send/receive operations are not yet optimized for the DGX-1 NVLink topology.

Fixed Issues

The following issues have been resolved in NCCL 2.7.6:

- Fixed crash when NVswitch is not visible inside a virtual machine.

Chapter 50. NCCL Release 2.7.5

This is the NCCL 2.7.5 release notes. For previous NCCL release notes, refer to the NCCL Archives.

Compatibility

NCCL 2.7.5 has been tested with the following:

- Deep learning framework containers. Refer to the Support Matrix for the supported container version.
- This NCCL release supports CUDA 10.1, CUDA 10.2, and CUDA 11.0.

Known Issues

Send/receive operations have a number of limitations:

- Using send/receive operations in combination to launch work on multiple GPUs from a single process can fail or hang if the GPUs process different amounts of data. Setting `NCCL_LAUNCH_MODE=PARALLEL` can work around the issue, but can also cause other problems. For more information, see the NCCL User Guide section Troubleshooting > Known Issues > Concurrency Between NCCL and CUDA calls.
- Aggregation is not supported on a given source-destination pair, meaning that there can only be one send/receive per source-destination pair.
- Each source-destination pair allocates a dedicated fifo of 4 MB (see `NCCL_BUFFSIZE` variable) which can use a significant amount of GPU memory at scale.
- When using GPU Direct RDMA, each point-to-point connection will also use resources on the GPU PCI address space, which we might run out of on some models. If that happens, consider disabling GPU Direct RDMA (`NCCL_NET_GDR_LEVEL=0`) or reduce the per-peer buffer size (`NCCL_BUFFSIZE`).
- Send/receive operations are not yet optimized for the DGX-1 NVLink topology.
- Running inside a virtual machine on an NVswitch platform can cause a crash if NVswitch is not visible inside the virtual machine.

Fixed Issues

The following issues have been resolved in NCCL 2.7.5:

- Minor fixes for A100 platforms.
- Add proper message for invalid `GroupEnd` call.

Chapter 51. NCCL Release 2.7.3

This is the NCCL 2.7.3 release notes. For previous NCCL release notes, refer to the NCCL Archives.

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- Added support for A100 GPU and related platforms
- Added support for CUDA 11
- Added support for send/receive operations (beta)

Compatibility

NCCL 2.7.3 has been tested with the following:

- Deep learning framework containers. Refer to the Support Matrix for the supported container version.
- This NCCL release supports CUDA 10.1, CUDA 10.2, and CUDA 11.0.

Known Issues

Send/receive operations have a number of limitations:

- Using send/receive operations in combination to launch work on multiple GPUs from a single process can fail or hang if the GPUs process different amounts of data. Setting `NCCL_LAUNCH_MODE=PARALLEL` can work around the issue, but can also cause other problems. For more information, see the NCCL User Guide section Troubleshooting > Known Issues > Concurrency Between NCCL and CUDA calls.
- Aggregation is not supported on a given source-destination pair, meaning that there can only be one send/receive per source-destination pair.
- Each source-destination pair allocates a dedicated fifo of 4 MB (see `NCCL_BUFFSIZE` variable) which can use a significant amount of GPU memory at scale.
- When using GPU Direct RDMA, each point-to-point connection will also use resources on the GPU PCI address space, which we might run out of on some models. If that

happens, consider disabling GPU Direct RDMA (`NCCL_NET_GDR_LEVEL=0`) or reduce the per-peer buffer size (`NCCL_BUFFSIZE`).

- Send/receive operations are not yet optimized for the DGX-1 NVLink topology.

Fixed Issues

The following issues have been resolved in NCCL 2.7.3:

- Fixed crash when only a subset of GPUs are visible within a container (#326).

Chapter 52. NCCL Release 2.6.4

This is the NCCL 2.6.4 release notes. For previous NCCL release notes, refer to the NCCL Archives.

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- Added support for in-network collectives (beta)
- Added support for adaptive routing on Infiniband
- Added support for topology dump/injection through XML files
- Added speed detection for PCI, Infiniband and Ethernet cards
- Added CPU detection for AMD CPUs and ARM

Chapter 53. NCCL Release 2.5.6

This is the NCCL 2.5.6 release notes. For previous NCCL release notes, refer to the NCCL Archives.

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- Added new protocol to improve performance on small operations.
- Improved topology detection and tree/ring creation (#179, #262).
- Improved multi-node tree performance by sending/receiving from different GPUs.
- Added model-based tuning to switch between the different algorithms and protocols.
- Reworked P2P/SHM detection in containers (#155, #248).
- Added detection for duplicate CUDA devices and return an error (#231).
- Added tuning for Google Cloud's gVNIC platform.

Compatibility

NCCL 2.5.6 has been tested with the following:

- Deep learning framework containers. Refer to the Support Matrix for the supported container version.
- This NCCL release supports CUDA 9.0, CUDA 10.0, CUDA 10.1, and CUDA 10.2.

Fixed Issues

The following issues have been resolved in NCCL 2.5.6:

- Sporadic NCCL error "ring 0 does not loop back to start" (#179).
- NCCL doesn't form proper rings on GCP V100s (#262).

Chapter 54. NCCL Release 2.4.8

This is the NCCL 2.4.8 release notes. This release includes fixes from the previous NCCL 2.4.x releases as well as the following additional changes. For previous NCCL release notes, see the archived NCCL Release Notes.

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- Improved socket transport performance by splitting transfer over multiple sockets.

> Note: This feature adds two new environment variables `NCCL_SOCKET_NTHREADS` and `NCCL_NSOCKS_PERTHREAD` for users to tune NCCL performance on socket-based networks. See the NCCL documentation for more details.

Compatibility

NCCL 2.4.8 has been tested with the following:

- Deep learning framework 19.05 containers
- This NCCL release supports; CUDA 9.0, CUDA 9.2 , CUDA 10.0, and CUDA 10.1.

Fixed Issues

The following issues have been resolved in NCCL 2.4.8:

- Suboptimal performance with TCP over high bandwidth networks. (GitHub issue #209)

Known Issues

- On single node Power systems with 4 GPUs, some performance regressions have been observed compared to NCCL 2.4.2. These will be addressed in future NCCL releases.
- By default, NCCL does not enable direct P2P communication through different PCIe root ports on Intel Skylake CPU and later. This is due to a known performance issue when using P2P on these CPU versions. There is now a new BIOS and performance

tuning option available (PCIe Peer-to-Peer Serialization) from Intel and their OEM vendors that resolves this P2P bandwidth issue. If the BIOS performance tuning option has been enabled, then NCCL direct P2P connections can be re-enabled by setting `NCCL_P2P_LEVEL=5`.

Chapter 55. NCCL Release 2.4.7

This is the NCCL 2.4.7 release notes. This release includes fixes from the previous NCCL 2.4.x releases as well as the following additional changes. For previous NCCL release notes, see the archived NCCL Release Notes.

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- Improved bootstrap socket connection reliability at scale.
- Added detection of IBM/Power NVLink bridge device.
- Added NUMA support to PCI-E distance calculations on x86 architectures.

> Note: This adds a new level (5) for the `NCCL_P2P_LEVEL` and `NCCL_NET_GDR_LEVEL` environment variables. See the NCCL documentation for more details.

- Added the `NCCL_IGNORE_CPU_AFFINITY` environment variable.

Compatibility

NCCL 2.4.7 has been tested with the following:

- Deep learning framework 19.04 containers
- This NCCL release supports; CUDA 9.0, CUDA 9.2 , CUDA 10.0, and CUDA 10.1.

Fixed Issues

The following issues have been resolved in NCCL 2.4.7:

- Fixed hostname hashing issue. (GitHub issue #187)
- Fixed memory leaks. (GitHub issue #180)
- Fixed compiler warning. (GitHub issue #178)
- Replaced non-standard variable length arrays. (GitHub issue #171)
- Fixed Tree and Shared Memory crash. (GitHub PR #185)
- Fixed hangs during long running jobs.
- Fixed the `NCCL_RINGS` environment variable handling.

- Added extra checks to catch duplicate calls to `ncclCommDestroy()`. (GitHub issue #191)

Known Issues

- On single node Power systems with 4 GPUs, some performance regressions have been observed compared to NCCL 2.4.2. These will be addressed in future NCCL releases.
- By default, NCCL does not enable direct P2P communication through different PCIe root ports on Intel Skylake CPU and later. This is due to a known performance issue when using P2P on these CPU versions. There is now a new BIOS and performance tuning option available (PCIe Peer-to-Peer Serialization) from Intel and their OEM vendors that resolves this P2P bandwidth issue. If the BIOS performance tuning option has been enabled, then NCCL direct P2P connections can be re-enabled by setting `NCCL_P2P_LEVEL=5`.

Chapter 56. NCCL Release 2.4.2

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- Implemented tree-based algorithms for better All Reduce performance at scale and with small and medium size messages.
- Support for external network plugins (e.g., `libfabric`).
- Add `ncclCommGetAsyncError()` function to report errors happening during collective operations.
- Add `ncclCommAbort()` function to destroy a communicator, aborting any outstanding operations.
- Support different ranks having a different `CUDA_VISIBLE_DEVICES`.
- Add a best-effort mechanism to check for size mismatch among collective calls.

Fixed Issues

- Support communication between Mesos containers (Github issue #155).
- Fix case where `posix_fallocate()` returns EINTR (Github issue #137).
- NCCL threads no longer escape the CPU affinity set by the user or job scheduler.

Chapter 57. NCCL Release 2.3.7

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- Minor tuning of the LL threshold for multi-node jobs.
- Improve performance of initialization on large number of ranks.

Fixed Issues

- Fixed issue causing "WARN : Message truncated" errors.

Chapter 58. NCCL Release 2.3.5

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- This release is open-sourced with no new features or fixes.

Chapter 59. NCCL Release 2.3.4

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- Improve performance tuning on large number of ranks.
- Add `NCCL_P2P_LEVEL` and `NCCL_IB_GDR_LEVEL` knobs to finely control when to use GPU Direct P2P and GPU Direct RDMA.
- Reduce setup time for large scale jobs.
- Increased maximum number of rings supported to 16.
- Added a runtime NCCL version API: `ncclGetVersion()`.
- Added NCCL_DEBUG_SUBSYS to allow filtering of `NCCL_DEBUG=INFO` logging from different subsystems.
- Support for Turing based systems.

Fixed Issues

- Fix hang on Power platforms.
- Fix low inter-node bandwidth issue on multi-DGX2 systems.
- Fix crash when used with PID isolator.

Chapter 60. NCCL Release 2.2.13

Key Features and Enhancements

There were no new features and enhancements in this release.

Using NCCL 2.2.13

Ensure you are familiar with the following notes when using this release.

- If NCCL returns an error code, set the environment variable `NCCL_DEBUG` to `WARN` to receive an explicit error message.
- The NCCL 2.x API is different from NCCL 1.x. Some porting may be needed for NCCL 1.x applications to work correctly. Refer to the migration documentation in the NCCL Developer Guide.
- Starting in 2.2, NCCL supports collective aggregation. You can put multiple NCCL collective operations in between `ncclGroupStart()` and `ncclGroupEnd()` to enable this feature.

Known Issues

- Using multiple processes in conjunction with multiple threads to manage the different GPUs may in some cases cause `ncclCommInitRank` to fail while establishing IPCs (`cudaIpcOpenMemHandle`). This problem does not appear when using only processes or only threads. This issue is fixed in recent driver versions, therefore, consider updating to the latest driver.
- NCCL uses CUDA 9 cooperative group launch by default, which may induce increased latencies in multi-threaded programs. See the `NCCL_LAUNCH_MODE` knob to restore the original behavior.
- Driver version 390 can cause data corruption when used together with GPU Direct RDMA. Disabling GPU Direct RDMA by setting `NCCL_IB_CUDA_SUPPORT=0`, or upgrading to 396.26 or newer driver should resolve the issue.

Fixed Issues

- Fix crash in child processes after calling `ncclCommDestroy`.

Chapter 61. NCCL Release 2.2.12

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- Added support for collective operations aggregation.
- Added `ncclBroadcast` function.

Using NCCL 2.2.12

Ensure you are familiar with the following notes when using this release.

- If NCCL returns an error code, set the environment variable `NCCL_DEBUG` to `WARN` to receive an explicit error message.
- The NCCL 2.x API is different from NCCL 1.x. Some porting may be needed for NCCL 1.x applications to work correctly. Refer to the migration documentation in the NCCL Developer Guide.
- Starting in 2.2, NCCL supports collective aggregation. You can put multiple NCCL collective operations in between `ncclGroupStart()` and `ncclGroupEnd()` to enable this feature.

Known Issues

- Using multiple processes in conjunction with multiple threads to manage the different GPUs may in some cases cause `ncclCommInitRank` to fail while establishing IPCs (`cudaIpcOpenMemHandle`). This problem does not appear when using only processes or only threads. This issue is fixed in recent driver versions, therefore, consider updating to the latest driver.
- NCCL uses CUDA 9 cooperative group launch by default, which may induce increased latencies in multi-threaded programs. See the `NCCL_LAUNCH_MODE` knob to restore the original behavior.
- Driver version 390 and later can cause data corruption when used together with GPU Direct RDMA. Disabling GPU Direct RDMA by setting `NCCL_IB_CUDA_SUPPORT=0` or reverting to driver 387 should resolve the issue.

Fixed Issues

- No longer clear the CPU affinity during initialization functions
- Fix various large scale issues
- Reduce the size of the library
- Fix crash or hang with PyTorch relative to the usage of calls to fork

Chapter 62. NCCL Release 2.1.15

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- Added a variable to control InfiniBand Traffic Class and Retry Count.

Using NCCL 2.1.15

Ensure you are familiar with the following notes when using this release.

- The NCCL 2.x API is different from NCCL 1.x. Some porting may be needed for NCCL 1.x applications to work correctly. Refer to the migration documentation in the NCCL Developer Guide.
- Starting in 2.2, NCCL supports collective aggregation. You can put multiple collectives in between `ncclGroupStart()` and `ncclGroupEnd()` to enable this feature.

Known Issues

- If NCCL returns an error code, set the environment variable `NCCL_DEBUG` to `WARN` to receive an explicit error message.
- Using multiple processes in conjunction with multiple threads to manage the different GPUs may in some cases cause `ncclCommInitRank` to fail while establishing IPCs (`cudaIpcOpenMemHandle`). This problem does not appear when using only processes or only threads.
- NCCL uses CUDA 9 cooperative group launch by default, which may induce increased latencies in multi-threaded programs. See the `NCCL_LAUNCH_MODE` knob to restore the original behavior.

Fixed Issues

- Fixed CPU usage and scheduling of NCCL network threads.
- Fixed CUDA launch crash when mixing different types of GPUs in a node.
- Fixed a performance problem on Skylake CPUs.
- Fixed hanging issues with cudaFree and inter-node communication.

- Restored library installation path to `/usr/lib/x86_64-linux-gnu` in debian packages.
- Fixed RoCEv2 failure when using a non-zero GID.
- No longer link to `stdc++` library statically as this can cause issues with C++ applications.
- Fixed PyTorch hanging issues when using multiple rings and many back-to-back broadcast operations.

Chapter 63. NCCL Release 2.1.4

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- Added support for InfiniBand GID selection, enabling the use of RoCE v2.
- Added support for InfiniBand Service Level (SL) selection.

Using NCCL 2.1.4

Ensure you are familiar with the following notes when using this release.

- The NCCL 2.x API is different from NCCL 1.x. Some porting may be needed for NCCL 1.x applications to work correctly. Refer to the migration documentation in the NCCL Developer Guide.

Known Issues

- If NCCL returns an error code, set the environment variable `NCCL_DEBUG to WARN` to receive an explicit error message.
- Using multiple processes in conjunction with multiple threads to manage the different GPUs may in some cases cause `ncclCommInitRank` to fail while establishing IPCs (`cudaIpcOpenMemHandle`). This problem does not appear when using only processes or only threads.
- NCCL uses CUDA® 9 cooperative group launch by default, which may induce increased latencies in multi-threaded programs. See the `NCCL_LAUNCH_MODE` knob in the NCCL Developer Guide to restore the original behavior.
- NCCL 2.1.4-1 embeds `libstdc++` and exports its symbols. This can break C++ applications.

Fixed Issues

- Fixed bug causing CUDA IPC to fail in some situations.
- Fixed bug causing a crash when p2p mappings are exhausted instead of returning an error.

Chapter 64. NCCL Release 2.1.2

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- New algorithms for improved latency communication
- RoCE support

Using NCCL 2.1.2

Ensure you are familiar with the following notes when using this release.

- The NCCL 2.x API is different from NCCL 1.x. Some porting may be needed for NCCL 1.x applications to work correctly. Refer to the migration documentation in the NCCL Developer Guide.

Known Issues

- If NCCL returns an error code, set the environment variable `NCCL_DEBUG to WARN` to receive an explicit error message.
- Using multiple processes in conjunction with multiple threads to manage the different GPUs may in some cases cause `ncclCommInitRank` to fail while establishing IPCs (`cudaIpcOpenMemHandle`). This problem does not appear when using only processes or only threads.
- NCCL uses CUDA 9 cooperative group launch by default, which may induce increased latencies in multi-threaded programs. See the `NCCL_LAUNCH_MODE` knob in the NCCL Developer Guide to restore the original behavior.

Fixed Issues

- NCCL now uses CUDA 9 cooperative groups to launch the CUDA kernels, fixing a long-standing issue with CUDA and NCCL operations being interleaved and potentially causing hangs.

Chapter 65. NCCL Release 2.0.5

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- NCCL 2.0.5 provides support for intra-node and inter-node communication.
- NCCL optimizes intra-node communication using NVLink, PCI express, and shared memory.
- Between nodes, NCCL implements fast transfers over sockets or InfiniBand verbs.
- GPU-to-GPU and GPU-to-Network direct transfers, using the GPU Direct technology, are extensively used when the hardware topology permits it.

Using NCCL 2.0.5

Ensure you are familiar with the following notes when using this release.

- The NCCL 2.0 API is different from NCCL 1.x. Some porting may be needed for NCCL 1.x applications to work correctly. Refer to the migration documentation in the NCCL Developer Guide.
- NCCL 2.0.5 has the new configuration file support. The NCCL environment variables can now be set in `~/.nccl.conf` and `/etc/nccl.conf`.
- Values defined in `~/.nccl.conf` take precedence over ones in `/etc/nccl.conf`.
- The syntax for each line of the NCCL configuration file is `<NCCL_VAR_NAME>=<VALUE>`.

Known Issues

- If NCCL returns any error code, set the environment variable `NCCL_DEBUG` to `WARN` to receive an explicit error message.
- RoCE support is experimental.
- Using multiple processes in conjunction with multiple threads to manage the different GPUs may in some cases cause `ncclCommInitRank` to fail while establishing IPCs (`cudaIpcOpenMemHandle`). This problem does not appear when using only processes or only threads.

Chapter 66. NCCL Release 2.0.4

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- NCCL 2.0.4 provides support for intra-node and inter-node communication.
- NCCL optimizes intra-node communication using NVLink, PCI express, and shared memory.
- Between nodes, NCCL implements fast transfers over sockets or InfiniBand verbs.
- GPU-to-GPU and GPU-to-Network direct transfers, using the GPU Direct technology, are extensively used when the hardware topology permits it.

Using NCCL 2.0.4

Ensure you are familiar with the following notes when using this release.

- The NCCL 2.0 API is different from NCCL 1.x. Some porting may be needed for NCCL 1.x applications to work correctly. Refer to the migration documentation in the NCCL Developer Guide.
- NCCL 2.0.4 has the new configuration file support. The NCCL environment variables can now be set in `~/.nccl.conf` and `/etc/nccl.conf`.
- Values defined in `~/.nccl.conf` take precedence over ones in `/etc/nccl.conf`.
- The syntax for each line of the NCCL configuration file is `<NCCL_VAR_NAME>=<VALUE>`.

Known Issues

- If NCCL returns any error code, set the environment variable `NCCL_DEBUG` to `WARN` to receive an explicit error message.
- RoCE is not supported.
- Using multiple processes in conjunction with multiple threads to manage the different GPUs may in some cases cause `ncclCommInitRank` to fail while establishing IPCs (`cudaIpcOpenMemHandle`). This problem does not appear when using only processes or only threads.

Chapter 67. NCCL Release 2.0.2

Key Features and Enhancements

This NCCL release includes the following key features and enhancements.

- NCCL 2.0.2 provides support for intra-node and inter-node communication.
- NCCL optimizes intra-node communication using NVLink, PCI express, and shared memory.
- Between nodes, NCCL implements fast transfers over sockets or InfiniBand verbs.
- GPU-to-GPU and GPU-to-Network direct transfers, using the GPU Direct technology, is extensively used when the hardware topology permits it.

Using NCCL 2.0.2

Ensure you are familiar with the following notes when using this release.

- The NCCL 2.0 API is different from NCCL 1.x. Some porting may be needed for NCCL 1.x applications to work correctly. Refer to the migration documentation in the NCCL Developer Guide.

Known Issues

- NCCL 2.0.2 is known to not work with CUDA driver 384.40 and later.
- If NCCL returns any error code, set the environment variable `NCCL_DEBUG` to `WARN` to receive an explicit error message.
- NCCL 2.0.2 does not support RoCE, that is, InfiniBand cards using Ethernet as link layer. The presence of an RoCE card on a node will make NCCL fail even when run within the node.
- Using multiple processes in conjunction with multiple threads to manage the different GPUs may in some cases cause `ncclCommInitRank` to fail while establishing IPCs (`cudaIpcOpenMemHandle`). This problem does not appear when using only processes or only threads.

Notice

This document is provided for information purposes only and shall not be regarded as a warranty of a certain functionality, condition, or quality of a product. NVIDIA Corporation ("NVIDIA") makes no representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assumes no responsibility for any errors contained herein. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This document is not a commitment to develop, release, or deliver any Material (defined below), code, or functionality.

NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice.

Customer should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgment, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer ("Terms of Sale"). NVIDIA hereby expressly objects to applying any customer general terms and conditions with regards to the purchase of the NVIDIA product referenced in this document. No contractual obligations are formed either directly or indirectly by this document.

NVIDIA products are not designed, authorized, or warranted to be suitable for use in medical, military, aircraft, space, or life support equipment, nor in applications where failure or malfunction of the NVIDIA product can reasonably be expected to result in personal injury, death, or property or environmental damage. NVIDIA accepts no liability for inclusion and/or use of NVIDIA products in such equipment or applications and therefore such inclusion and/or use is at customer's own risk.

NVIDIA makes no representation or warranty that products based on this document will be suitable for any specified use. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer's sole responsibility to evaluate and determine the applicability of any information contained in this document, ensure the product is suitable and fit for the application planned by customer, and perform the necessary testing for the application in order to avoid a default of the application or the product. Weaknesses in customer's product designs may affect the quality and reliability of the NVIDIA product and may result in additional or different conditions and/or requirements beyond those contained in this document. NVIDIA accepts no liability related to any default, damage, costs, or problem which may be based on or attributable to: (i) the use of the NVIDIA product in any manner that is contrary to this document or (ii) customer product designs.

Trademarks

NVIDIA, the NVIDIA logo, and cuBLAS, CUDA, CUDA Toolkit, cuDNN, DALI, DIGITS, DGX, DGX-1, DGX-2, DGX Station, DLProf, GPU, Jetson, Kepler, Maxwell, NCCL, Nsight Compute, Nsight Systems, NVCaffe, NVIDIA Deep Learning SDK, NVIDIA Developer Program, NVIDIA GPU Cloud, NVLink, NVSHMEM, PerfWorks, Pascal, SDK Manager, Tegra, TensorRT, TensorRT Inference Server, Tesla, TF-TRT, Triton Inference Server, Turing, and Volta are trademarks and/or registered trademarks of NVIDIA Corporation in the United States and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright

© 2026 NVIDIA CORPORATION and affiliates. All rights reserved.