Release Notes

Release 12.3

NVIDIA

Oct 20, 2023

Contents

NVIDIA CUDA Toolkit Release Notes

The Release Notes for the CUDA Toolkit.

The release notes for the NVIDIA® CUDA® Toolkit can be found online at https://docs.nvidia.com/cuda/cuda-toolkit-release-notes/index.html.

Note: The release notes have been reorganized into two major sections: the general CUDA release notes, and the CUDA libraries release notes including historical information for 12.x releases.

Chapter 1. CUDA Toolkit Major Component Versions

CUDA Components Starting with CUDA 11, the various components in the toolkit are versioned independently.

For CUDA 12.3, the table below indicates the versions:

Table 1: CUDA 12.3 Component Versions

Component Name		Version Information	Supported Architectures	Supported Platforms
CUDA C++ Core Compute Libraries	Thrust	2.2.0	x86_64, arm64-sbsa, POWER	Linux, Windows
	CUB	2.2.0		
	libcu++	2.2.0		
	Cooperative Groups	12.3.0		
CUDA Runtime (cudart)		12.3.52	x86_64, arm64-sbsa, POWER	Linux, Windows, WSL
cuobjdump		12.3.52	x86_64, arm64-sbsa, POWER	Linux, Windows
CUPTI		12.3.52	x86_64, arm64-sbsa, POWER	Linux, Windows, WSL
CUDA cuxxfilt (demangler)		12.3.52	x86_64, arm64-sbsa, POWER	Linux, Windows
CUDA Demo Suite		12.3.52	x86_64	Linux, Windows
CUDA GDB		12.3.52	x86_64, arm64-sbsa, POWER	Linux, WSL
CUDA Nsight Eclipse Plugin		12.3.52	x86_64, POWER	Linux
CUDA NVCC		12.3.52	x86_64, arm64-sbsa, POWER	Linux, Windows, WSL
CUDA nvdisasm		12.3.52	x86_64, arm64-sbsa, POWER	Linux, Windows

continues on next page

Table 1 – continued from previous page

Component Name	Version Information	Supported Architectures	Supported Platforms
CUDA NVML Headers	12.3.52	x86_64, arm64-sbsa, POWER	Linux, Windows, WSL
CUDA nvprof	12.3.52	x86_64, POWER	Linux, Windows
CUDA nvprune	12.3.52	x86_64, arm64-sbsa, POWER	Linux, Windows, WSL
CUDA NVRTC	12.3.52	x86_64, arm64-sbsa, POWER	Linux, Windows, WSL
NVTX	12.3.52	x86_64, arm64-sbsa, POWER	Linux, Windows, WSL
CUDA NVVP	12.3.52	x86_64, POWER	Linux, Windows
CUDA OpenCL	12.3.52	x86_64	Linux, Windows
CUDA Profiler API	12.3.52	x86_64, arm64-sbsa, POWER	Linux, Windows, WSL
CUDA Compute Sanitizer API	12.3.52	x86_64, arm64-sbsa, POWER	Linux, Windows, WSL
CUDA cuBLAS	12.3.2.9	x86_64, arm64-sbsa, POWER	Linux, Windows, WSL
CUDA cuFFT	11.0.11.19	x86_64, arm64-sbsa, POWER	Linux, Windows, WSL
CUDA cuFile	1.8.0.34	x86_64, arm64-sbsa	Linux
CUDA cuRAND	10.3.4.52	x86_64, arm64-sbsa, POWER	Linux, Windows, WSL
CUDA cuSOLVER	11.5.3.52	x86_64, arm64-sbsa, POWER	Linux, Windows, WSL
CUDA cuSPARSE	12.1.3.153	x86_64, arm64-sbsa, POWER	Linux, Windows, WSL
CUDA NPP	12.2.2.32	x86_64, Parm64-sbsa, OWER	Linux, Windows, WSL
CUDA nvJitLink	12.3.52	x86_64, arm64-sbsa, POWER	Linux, Windows, WSL
CUDA nvJPEG	12.2.3.32	x86_64, arm64-sbsa, POWER	Linux, Windows, WSL
Nsight Compute	2023.3.0.12	x86_64, arm64-sbsa, POWER	Linux, Windows, WSL (Windows 11)
Nsight Systems	2023.3.3.42	x86_64, arm64-sbsa, POWER	Linux, Windows, WSL

continues on next page

Table 1 – continued from previous page

Component Name	Version Information	Supported Architectures	Supported Platforms
Nsight Visual Studio Edition (VSE)	2023.2.3.23248	x86_64 (Windows)	Windows
nvidia_fs[1]	2.18.3	x86_64, arm64-sbsa	Linux
Visual Studio Integration	12.3.52	x86_64 (Windows)	Windows
NVIDIA Linux Driver	545.23.06	x86_64, arm64-sbsa, POWER	Linux
NVIDIA Windows Driver	545.84	x86_64 (Windows)	Windows, WSL

CUDA Driver Running a CUDA application requires the system with at least one CUDA capable GPU and a driver that is compatible with the CUDA Toolkit. See Table 3. For more information various GPU products that are CUDA capable, visit https://developer.nvidia.com/cuda-gpus.

Each release of the CUDA Toolkit requires a minimum version of the CUDA driver. The CUDA driver is backward compatible, meaning that applications compiled against a particular version of the CUDA will continue to work on subsequent (later) driver releases.

More information on compatibility can be found at https://docs.nvidia.com/cuda/cuda-c-best-practices-guide/index.html#cuda-compatibility-and-upgrades.

Note: Starting with CUDA 11.0, the toolkit components are individually versioned, and the toolkit itself is versioned as shown in the table below.

The minimum required driver version for CUDA minor version compatibility is shown below. CUDA minor version compatibility is described in detail in https://docs.nvidia.com/deploy/cuda-compatibility/index.html

[1] Only available on select Linux distros

Table 2: CUDA Toolkit and Minimum Required Driver Version for CUDA Minor Version Compatibility

CUDA Toolkit	**Minimum Required Driver Version for CUDA Minor Version Compatibility***	
	Linux x86_64 Driver Version	Windows x86_64 Driver Version
CUDA 12.3.x	>=525.60.13	>=527.41
CUDA 12.2.x	>=525.60.13	>=527.41
CUDA 12.1.x	>=525.60.13	>=527.41
CUDA 12.0.x	>=525.60.13	>=527.41
CUDA 11.8.x	>=450.80.02	>=452.39
CUDA 11.7.x	>=450.80.02	>=452.39
CUDA 11.6.x	>=450.80.02	>=452.39
CUDA 11.5.x	>=450.80.02	>=452.39
CUDA 11.4.x	>=450.80.02	>=452.39
CUDA 11.3.x	>=450.80.02	>=452.39
CUDA 11.2.x	>=450.80.02	>=452.39
CUDA 11.1 (11.1.0)	>=450.80.02	>=452.39
CUDA 11.0 (11.0.3)	>=450.36.06**	>=451.22**

* Using a Minimum Required Version that is **different** from Toolkit Driver Version could be allowed in compatibility mode – please read the CUDA Compatibility Guide for details.

** CUDA 11.0 was released with an earlier driver version, but by upgrading to Tesla Recommended Drivers 450.80.02 (Linux) / 452.39 (Windows), minor version compatibility is possible across the CUDA 11.x family of toolkits.

The version of the development NVIDIA GPU Driver packaged in each CUDA Toolkit release is shown below.

Table 3: CUDA Toolkit and Corresponding Driver Versions

CUDA Toolkit	**Toolkit Driver Version**	
	Linux x86_64 Driver Version	Windows x86_64 Driver Version
CUDA 12.3 GA	>=545.23.06	>=545.84
CUDA 12.2 Update 2	>=535.104.05	>=537.13
CUDA 12.2 Update 1	>=535.86.09	>=536.67
CUDA 12.2 GA	>=535.54.03	>=536.25
CUDA 12.1 Update 1	>=530.30.02	>=531.14
CUDA 12.1 GA	>=530.30.02	>=531.14
CUDA 12.0 Update 1	>=525.85.12	>=528.33

continues on next page

Table 3 – continued from previous page

CUDA Toolkit	Toolkit Driver Version	
CUDA 12.0 GA	>=525.60.13	>=527.41
CUDA 11.8 GA	>=520.61.05	>=520.06
CUDA 11.7 Update 1	>=515.48.07	>=516.31
CUDA 11.7 GA	>=515.43.04	>=516.01
CUDA 11.6 Update 2	>=510.47.03	>=511.65
CUDA 11.6 Update 1	>=510.47.03	>=511.65
CUDA 11.6 GA	>=510.39.01	>=511.23
CUDA 11.5 Update 2	>=495.29.05	>=496.13
CUDA 11.5 Update 1	>=495.29.05	>=496.13
CUDA 11.5 GA	>=495.29.05	>=496.04
CUDA 11.4 Update 4	>=470.82.01	>=472.50
CUDA 11.4 Update 3	>=470.82.01	>=472.50
CUDA 11.4 Update 2	>=470.57.02	>=471.41
CUDA 11.4 Update 1	>=470.57.02	>=471.41
CUDA 11.4.0 GA	>=470.42.01	>=471.11
CUDA 11.3.1 Update 1	>=465.19.01	>=465.89
CUDA 11.3.0 GA	>=465.19.01	>=465.89
CUDA 11.2.2 Update 2	>=460.32.03	>=461.33
CUDA 11.2.1 Update 1	>=460.32.03	>=461.09
CUDA 11.2.0 GA	>=460.27.03	>=460.82
CUDA 11.1.1 Update 1	>=455.32	>=456.81
CUDA 11.1 GA	>=455.23	>=456.38
CUDA 11.0.3 Update 1	>= 450.51.06	>= 451.82
CUDA 11.0.2 GA	>= 450.51.05	>= 451.48
CUDA 11.0.1 RC	>= 450.36.06	>= 451.22
CUDA 10.2.89	>= 440.33	>= 441.22
CUDA 10.1 (10.1.105 general release, and updates)	>= 418.39	>= 418.96
CUDA 10.0.130	>= 410.48	>= 411.31
CUDA 9.2 (9.2.148 Update 1)	>= 396.37	>= 398.26
CUDA 9.2 (9.2.88)	>= 396.26	>= 397.44
CUDA 9.1 (9.1.85)	>= 390.46	>= 391.29

continues on next page

Table 3 – continued from previous page

CUDA Toolkit	Toolkit Driver Version	
CUDA 9.0 (9.0.76)	>= 384.81	>= 385.54
CUDA 8.0 (8.0.61 GA2)	>= 375.26	>= 376.51
CUDA 8.0 (8.0.44)	>= 367.48	>= 369.30
CUDA 7.5 (7.5.16)	>= 352.31	>= 353.66
CUDA 7.0 (7.0.28)	>= 346.46	>= 347.62

For convenience, the NVIDIA driver is installed as part of the CUDA Toolkit installation. Note that this driver is for development purposes and is not recommended for use in production with Tesla GPUs.

For running CUDA applications in production with Tesla GPUs, it is recommended to download the latest driver for Tesla GPUs from the NVIDIA driver downloads site at https://www.nvidia.com/drivers.

During the installation of the CUDA Toolkit, the installation of the NVIDIA driver may be skipped on Windows (when using the interactive or silent installation) or on Linux (by using meta packages).

For more information on customizing the install process on Windows, see https://docs.nvidia.com/cuda/cuda-installation-guide-microsoft-windows/index.html#install-cuda-software.

For meta packages on Linux, see https://docs.nvidia.com/cuda/cuda-installation-guide-linux/index.html#package-manager-metas.

Chapter 2. New Features

This section lists new general CUDA and CUDA compilers features.

2.1. General CUDA

- CUDA User Mode Driver, CUDA Runtime libraries and CUBLAS now come with obfuscated symbol names and with frame pointers enabled.
- Frame Pointers are enabled for other libraries in the CUDA Toolkit: NVIDIA Management Library, CUDA Profiling Tools Interface, cuBLAS, Compiler libraries – NVRTC, PTXJIT compiler, nvJitLink, and libnvvm.
  - Allows better runtime visibility and traceability, and allows easier exchange of runtime infor-mation with NVIDIA when needed for debugging purposes.
  - See https://developer.nvidia.com/blog/cuda-toolkit-symbol-server/ for information on how to use obfuscated symbols.
  - Symbol server address is: https://cudatoolkit-symbols.nvidia.com/.
- Lazy loading default enablement for Windows:
  - Brings the significant memory savings and load-time reductions of lazy loading to Windows by default. Additionally, makes the behavior equivalent between Linux and Windows.
- Single-step CUDA uninstall for Windows:
  - It is no longer necessary to uninstall multiple components of the CUDA Toolkit individually to upgrade or uninstall CUDA. This can now be done in a single step.
- CUDA Graphs:
  - Graph edge data, allowing modified dependencies between nodes. Programmatic Dependent Launch may now be described natively in CUDA Graphs.
- Launch completion events:
  - Allows a dependency on scheduling, but not completion, of all blocks in a kernel, enabling tighter control of scheduling.
- MPS:
  - Added a CUDA API to query whether or not MPS is running.

- Added a driver API to return the name of a kernel function.
- Added an API to libnvJitLink to return the nvJitLink version.
- Added support for reading kernel parameters in device functions.
- Enable querying the return type of __device__ lambdas with trailing return type. Fixes uncommon failures when using device-side lambdas.
- NVML / nvidia-smi:
  - Metric for front-end context switch utilization (FECS)
  - Added metrics for Ada Lovelace AV1 codec utilization
  - Support GPU monitoring on Tegra
  - Added an NVML API to expose H100 PCIe counters and corresponding PCIe section in nvidia-smi

2.2. CUDA Compilers

- For changes to PTX, refer to https://docs.nvidia.com/cuda/parallel-thread-execution/#ptx-isa-version-8-3.
- Enhanced thread support when using the libNVVM API. Clients can take advantage of improved compilation speeds by spawning multiple compilation threads concurrently.
- Improved compile time in some common scenarios:
  - Extended split compilation to cubin for LTO.
  - Turned on concurrent NVVM processing by default, with documented fallback to serialized compilation.
  - Reduced NVRTC compile time for small programs via moving CUDA C++ builtin function declarations into compiler.
  - Moved `cuda_fp16.h` and `cuda_bf16.h` into compiler bitcode.
- Added new keyword``__inline_hint__`` to specify device functions in a different `.cu` file to be inlined during LTO.
- Enabled querying return type of __device__ lambdas with trailing return type.
- Provided information about unused bytes to compute-sanitizer for better diagnostics.

2.3. CUDA Developer Tools

- For changes to nvprof and Visual Profiler, see the changelog.
- For new features, improvements, and bug fixes in Nsight Systems, see the changelog.
- For new features, improvements, and bug fixes in Nsight Visual Studio Edition, see the changelog.
- For new features, improvements, and bug fixes in CUPTI, see the changelog.
- For new features, improvements, and bug fixes in Nsight Compute, see the changelog.

- For new features, improvements, and bug fixes in Compute Sanitizer, see the changelog.
- For new features, improvements, and bug fixes in CUDA-GDB, see the changelog.

Chapter 3. Resolved Issues

3.1. General CUDA

- Resolved an NVML incompatibility issue present when upgrading to driver version 535 without upgrading CUDA from 12.1 to 12.2.
- Improved driver error reporting in rare conditions when ECC errors impact GPU initialization.

Chapter 4. Deprecated or Dropped Features

Features deprecated in the current release of the CUDA software still work in the current release, but their documentation may have been removed, and they will become officially unsupported in a future release. We recommend that developers employ alternative solutions to these features in their software.

General CUDA

- Starting in CUDA 12.4, the NVIDIA driver installation on Linux will be opt-in. The goal is to improve user experience for a wide range of use cases such as installing the open module flavor driver. The `cuda-runtime` dependency and therefore the `cuda-drivers` (NVIDIA driver) dependency will be removed from the top-level `cuda` meta-package. Effectively, the `cuda` and `cuda-toolkit` meta-packages will be equivalent in CUDA 12.4.

CUDA Tools

- Support for the macOS host client of CUDA-GDB is deprecated. It will be dropped in an upcoming release.

Chapter 5. Known Issues

5.1. General CUDA Known Issues

- CUDA kernels that use the sparsity feature of tensor cores through the mma.sp *PTX instruction <https://docs.nvidia.com/cuda/parallel-thread-execution/index.html#warp-level-matrix-instructions-for-sparse-mma>*`__ on NVIDIA Hopper architecture GPUs may intermittently experience silent data corruption resulting in incorrect results. NVIDIA libraries currently do not provide access to tensor cores with sparsity so only kernels directly developed using the `mma.sp` PTX instruction are impacted. This issue will be fixed in an upcoming release.
- The Early Access (EA) of Hopper Confidential Computing is not enabled on 12.3 or its associated driver (545.xx). Please see https://docs.nvidia.com/confidential-computing/ for details.
- The aarch64-jetson architecture for Jetson devices is not supported in the CUDA 12.3 release.

Chapter 6. CUDA Libraries

This section covers CUDA Libraries release notes for 12.x releases.

- CUDA Math Libraries toolchain uses C++11 features, and a C++11-compatible standard library (libstdc++ >= 20150422) is required on the host.
- Support for the following compute capabilities is removed for all libraries:
  - sm_35 (Kepler)
  - sm_37 (Kepler)

6.1. cuBLAS Library

6.1.1. cuBLAS: Release 12.3

- **New Features**
  - Improved performance on NVIDIA L40S Ada GPUs.
- **Known Issues**
  - cuBLASLt matmul operations may compute the output incorrectly under the following conditions: the data type of matrices A and B is FP8, the data type of matrices C and D is FP32, FP16, or BF16, the beta value is 1.0, the C and D matrices are the same, the epilogue contains GELU activation function.
  - When an application compiled with cuBLASLt from CUDA Toolkit 12.2 update 1 or earlier runs with cuBLASLt from CUDA Toolkit 12.2 update 2 or later, matrix multiply descriptors initialized using `cublasLtMatmulDescInit()` may not respect attribute changes using `cublasLtMatmulDescSetAttribute()`. To workaround this issue, create the matrix multiply descriptor using `cublasLtMatmulDescCreate()` instead of `cublasLtMatmulDescInit()`. This will be fixed in an upcoming release.

6.1.2. cuBLAS: Release 12.2 Update 2

- **New Features**
  - cuBLASLt will now attempt to decompose problems that cannot be run by a single gemm kernel. It does this by partitioning the problem into smaller chunks and executing the gemm kernel multiple times. This improves functional coverage for very large m, n, or batch size cases and makes the transition from the cuBLAS API to the cuBLASLt API more reliable.
- **Known Issues**
  - cuBLASLt matmul operations may compute the output incorrectly under the following conditions: the data type of matrices A and B is FP8, the data type of matrices C and D is FP32, FP16, or BF16, the beta value is 1.0, the C and D matrices are the same, the epilogue contains GELU activation function.

6.1.3. cuBLAS: Release 12.2

- **Known Issues**
  - cuBLAS initialization fails on Hopper architecture GPUs when MPS is in use with `CUDA_MPS_ACTIVE_THREAD_PERCENTAGE` set to a value less than 100%. There is currently no workaround for this issue.
  - Some Hopper kernels produce incorrect results for batched matmuls with `CUBLASLT_EPILOGUE_RELU_BIAS` or `CUBLASLT_EPILOGUE_GELU_BIAS` and a non-zero `CUBLASLT_MATMUL_DESC_BIAS_BATCH_STRIDE`. The kernels apply the first batch's bias vector to all batches. This will be fixed in a future release.

6.1.4. cuBLAS: Release 12.1 Update 1

- **New Features**
  - Support for FP8 on NVIDIA Ada GPUs.
  - Improved performance on NVIDIA L4 Ada GPUs.
  - Introduced an API that instructs the cuBLASLt library to not use some CPU instructions. This is useful in some rare cases where certain CPU instructions used by cuBLASLt heuristics negatively impact CPU performance. Refer to https://docs.nvidia.com/cuda/cublas/index.html#disabling-cpu-instructions.
- **Known Issues**
  - When creating a matrix layout using the `cublasLtMatrixLayoutCreate()` function, the object pointed at by `cublasLtMatrixLayout_t` is smaller than `cublasLtMatrixLayoutOpaque_t` (but enough to hold the internal structure). As a result, the object should not be dereferenced or copied explicitly, as this might lead to out of bound accesses. If one needs to serialize the layout or copy it, it is recommended to manually allocate an object of size `sizeof(cublasLtMatrixLayoutOpaque_t)` bytes, and initialize it using `cublasLtMatrixLayoutInit()` function. The same applies to `cublasLtMatmulDesc_t` and `cublasLtMatrixTransformDesc_t`. The issue will be fixed in future releases by ensuring that `cublasLtMatrixLayoutCreate()` allocates at least `sizeof(cublasLtMatrixLayoutOpaque_t)` bytes.

6.1.5. cuBLAS: Release 12.0 Update 1

- **New Features**
  - Improved performance on NVIDIA H100 SXM and NVIDIA H100 PCIe GPUs.
- **Known Issues**
  - For optimal performance on NVIDIA Hopper architecture, cuBLAS needs to allocate a bigger internal workspace (64 MiB) than on the previous architectures (8 MiB). In the current and previous releases, cuBLAS allocates 256 MiB. This will be addressed in a future release. A possible workaround is to set the `CUBLAS_WORKSPACE_CONFIG` environment variable to :32768:2 when running cuBLAS on NVIDIA Hopper architecture.
- **Resolved Issues**
  - Reduced cuBLAS host-side overheads caused by not using the cublasLt heuristics cache. This began in the CUDA Toolkit 12.0 release.
  - Added forward compatible single precision complex GEMM that does not require workspace.

6.1.6. cuBLAS: Release 12.0

- **New Features**
  - `cublasLtMatmul` now supports FP8 with a non-zero beta.
  - Added `int64` APIs to enable larger problem sizes; refer to 64-bit integer interface.
  - Added more Hopper-specific kernels for `cublasLtMatmul` with epilogues:
    - `CUBLASLT_EPILOGUE_BGRAD{A,B}`
    - `CUBLASLT_EPILOGUE_{RELU,GELU}_AUX`
    - `CUBLASLT_EPILOGUE_D{RELU,GELU}`
  - Improved Hopper performance on arm64-sbsa by adding Hopper kernels that were previously supported only on the x86_64 architecture for Windows and Linux.
- **Known Issues**
  - There are no forward compatible kernels for single precision complex gemms that do not require workspace. Support will be added in a later release.
- **Resolved Issues**
  - Fixed an issue on NVIDIA Ampere architecture and newer GPUs where `cublasLtMatmul` with epilogue `CUBLASLT_EPILOGUE_BGRAD{A,B}` and a nontrivial reduction scheme (that is, not `CUBLASLT_REDUCTION_SCHEME_NONE`) could return incorrect results for the bias gradient.
  - `cublasLtMatmul` for gemv-like cases (that is, m or n equals 1) might ignore bias with the `CUBLASLT_EPILOGUE_RELU_BIAS` and `CUBLASLT_EPILOGUE_BIAS` epilogues.

  Deprecations
  - Disallow including `cublas.h` and `cublas_v2.h` in the same translation unit.
  - Removed:

- CUBLAS_MATMUL_STAGES_16x80 and CUBLAS_MATMUL_STAGES_64x80 from cublasLtMatmulStages_t. No kernels utilize these stages anymore.
- cublasLt3mMode_t, CUBLASLT_MATMUL_PREF_MATH_MODE_MASK, and CUBLASLT_MATMUL_PREF_GAUSSIAN_MODE_MASK from cublasLtMatmulPreferenceAttributes_t. Instead, use the corresponding flags from cublasLtNumericalImplFlags_t.
- CUBLASLT_MATMUL_PREF_POINTER_MODE_MASK, CUBLASLT_MATMUL_PREF_EPILOGUE_MASK, and CUBLASLT_MATMUL_PREF_SM_COUNT_TARGET from cublasLtMatmulPreferenceAttributes_t. The corresponding parameters are taken directly from cublasLtMatmulDesc_t.
- CUBLASLT_POINTER_MODE_MASK_NO_FILTERING from cublasLtPointerModeMask_t. This mask was only applicable to CUBLASLT_MATMUL_PREF_MATH_MODE_MASK which was removed.

6.2. cuFFT Library

6.2.1. cuFFT: Release 12.3

- **New Features**
  - Callback kernels are more relaxed in terms of resource usage, and will use fewer registers.
  - Improved accuracy for double precision prime and composite FFT sizes with factors larger than 127.
  - Slightly improved planning times for some FFT sizes.

6.2.2. cuFFT: Release 12.2

- **New Features**
  - cufftSetStream can be used in multi-GPU plans with a stream from any GPU context, instead of from the primary context of the first GPU listed in cufftXtSetGPUs.
  - Improved performance of 1000+ of FFTs of sizes ranging from 62 to 16380. The improved performance spans hundreds of single precision and double precision cases for FFTs with contiguous data layout, across multiple GPU architectures (from Maxwell to Hopper GPUs) via PTX JIT.
  - Reduced the size of the static libraries when compared to cuFFT in the 12.1 release.
- **Resolved Issues**
  - cuFFT no longer exhibits a race condition when threads simultaneously create and access plans with more than 1023 plans alive.
  - cuFFT no longer exhibits a race condition when multiple threads call cufftXtSetGPUs concurrently.

6.2.3. cuFFT: Release 12.1 Update 1

- **Known Issues**
  - cuFFT exhibits a race condition when one thread calls `cufftCreate` (or `cufftDestroy`) and another thread calls any API (except `cufftCreate` or `cufftDestroy`), and when the total number of plans alive exceeds 1023.
  - cuFFT exhibits a race condition when multiple threads call `cufftXtSetGPUs` concurrently on different plans.

6.2.4. cuFFT: Release 12.1

- **New Features**
  - Improved performance on Hopper GPUs for hundreds of FFTs of sizes ranging from 14 to 28800. The improved performance spans over 542 cases across single and double precision for FFTs with contiguous data layout.
- **Known Issues**
  - Starting from CUDA 11.8, CUDA Graphs are no longer supported for callback routines that load data in out-of-place mode transforms. An upcoming release will update the cuFFT callback implementation, removing this limitation. cuFFT deprecated callback functionality based on separate compiled device code in cuFFT 11.4.
- **Resolved Issues**
  - cuFFT no longer produces errors with compute-sanitizer at program exit if the CUDA context used at plan creation was destroyed prior to program exit.

6.2.5. cuFFT: Release 12.0 Update 1

- **Resolved Issues**
  - Scratch space requirements for multi-GPU, single-batch, 1D FFTs were reduced.

6.2.6. cuFFT: Release 12.0

- **New Features**
  - PTX JIT kernel compilation allowed the addition of many new accelerated cases for Maxwell, Pascal, Volta and Turing architectures.
- **Known Issues**
  - cuFFT plan generation time increases due to PTX JIT compiling. Refer to Plan Initialization TIme.
- **Resolved Issues**
  - cuFFT plans had an unintentional small memory overhead (of a few kB) per plan. This is resolved.

6.3. cuSOLVER Library

6.3.1. cuSOLVER: Release 12.2 Update 2

- **Resolved Issues**
  - Fixed an issue with `cusolverDn<t>gesvd()`, `cusolverDnGesvd()`, and `cusolverDnXgesvd()`, which could cause wrong results for matrices larger than 18918 if `jobu` or `jobvt` was unequal to 'N'.

6.3.2. cuSOLVER: Release 12.2

- **New Features**
  - A new API to ensure deterministic results or allow non-deterministic results for improved performance. See `cusolverDnSetDeterministicMode()` and `cusolverDnGetDeterministicMode()`. Affected functions are: `cusolverDn<t>geqrf()`, `cusolverDn<t>syevd()`, `cusolverDn<t>syevdx()`, `cusolverDn<t>gesvdj()`, `cusolverDnXgeqrf()`, `cusolverDnXsyevd()`, `cusolverDnXsyevdx()`, `cusolverDnXgesvdr()`, and `cusolverDnXgesvdp()`.
- **Known Issues**
  - Concurrent executions of `cusolverDn<t>getrf()` or `cusolverDnXgetrf()` in different non-blocking CUDA streams on the same device might result in a deadlock.

6.4. cuSPARSE Library

6.4.1. cuSPARSE: Release 12.3

- **New Features**
  - The `cusparseSpSV_bufferSize()` and `cusparseSpSV_analysis()` routines now accept NULL pointers for the dense vector.
  - The `cusparseSpSM_bufferSize()` and `cusparseSpSM_analysis()` routines now accept dense matrix descriptors with NULL pointer for values.
- **Known Issues**
  - The `cusparseSpSV_analysis()` and `cusparseSpSM_analysis()` routines are blocking calls/not asynchronous.
- **Resolved Issues**
  - `cusparseSpMV()` now supports output vector with the minimum alignment.
  - `cusparseSpSV()` provided indeterministic results in some cases.
  - Fixed an issue that caused `cusparseSpSV_analysis()` to hang sometimes in a multi-thread environment.

- Fixed an issue with `cusparseSpSV()` and `cusparseSpSV()` that somtimes yielded wrong output when the output vector/matrix or input matrix contained NaN.

6.4.2. cuSPARSE: Release 12.2 Update 1

- **New Features**
  - The library now provides the opportunity to dump sparse matrices to files during the creation of the descriptor for debugging purposes. See logging API https://docs.nvidia.com/cuda/cusparse/index.html#cusparse-logging-api.
- **Resolved Issues**
  - Removed `CUSPARSE_SPMM_CSR_ALG3` fallback to avoid confusion in the algorithm selection process.
  - Clarified the supported operations for `cusparseSDDMM()`.
  - `cusparseCreateConstSlicedEll()` now uses `const` pointers.
  - Fixed wrong results in rare edge cases of `cusparseCsr2CscEx2()` with base 1 indexing.
  - `cusparseSpSM_bufferSize()` could ask slightly less memory than needed.
  - `cusparseSpMV()` now checks the validity of the buffer pointer only when it is strictly needed.
- **Deprecations**
  - Several legacy APIs have been officially deprecated. A compile-time warning has been added to all of them.

6.4.3. cuSPARSE: Release 12.1 Update 1

- **New Features**
  - Introduced Block Sparse Row (BSR) sparse matrix storage for the Generic APIs with support for SDDMM routine (`cusparseSDDMM`).
  - Introduced Sliced Ellpack (SELL) sparse matrix storage format for the Generic APIs with support for sparse matrix-vector multiplication (`cusparseSpMV`) and triangular solver with a single right-hand side (`cusparseSpSV`).
  - Added a new API call (`cusparseSpSV_updateMatrix`) to update matrix values and/or the matrix diagonal in the sparse triangular solver with a single right-hand side after the analysis step.

6.4.4. cuSPARSE: Release 12.0 Update 1

- **New Features**
  - cusparseSDDMM() now supports mixed precision computation.
  - Improved cusparseSpMM() alg2 mixed-precision performance on some matrices on NVIDIA Ampere architecture GPUs.
  - Improved cusparseSpMV() performance with a new load balancing algorithm.
  - cusparseSpSV() and cusparseSpSM() now support in-place computation, namely the output and input vectors/matrices have the same memory address.
- **Resolved Issues**
  - cusparseSpSM() could produce wrong results if the leading dimension (ld) of the RHS matrix is greater than the number of columns/rows.

6.4.5. cuSPARSE: Release 12.0

- **New Features**
  - JIT LTO functionalities (cusparseSpMMOp()) switched from driver to nvJitLto library. Starting from CUDA 12.0 the user needs to link to libnvJitLto.so, see cuSPARSE documentation. JIT LTO performance has also been improved for cusparseSpMMOpPlan().
  - Introduced const descriptors for the Generic APIs, for example, cusparseConstSpVecGet(). Now the Generic APIs interface clearly declares when a descriptor and its data are modified by the cuSPARSE functions.
  - Added two new algorithms to cusparseSpGEMM() with lower memory utilization. The first algorithm computes a strict bound on the number of intermediate product, while the second one allows partitioning the computation in chunks.
  - Added int8_t support to cusparseGather(), cusparseScatter(), and cusparseCsr2cscEx2().
  - Improved cusparseSpSV() performance for both the analysis and the solving phases.
  - Improved cusparseSpSM() performance for both the analysis and the solving phases.
  - Improved cusparseSDDMM() performance and added support for batch computation.
  - Improved cusparseCsr2cscEx2() performance.
- **Resolved Issues**
  - cusparseSpSV() and cusparseSpSM() could produce wrong results.
  - cusparseDnMatGetStridedBatch() did not accept batchStride == 0.
- **Deprecations**
  - Removed deprecated CUDA 11.x APIs, enumerators, and descriptors.

6.5. Math Library

6.5.1. CUDA Math: Release 12.3

- **New Features**
  - Performance of SIMD Integer CUDA Math APIs was improved.
- **Known Issues**
  - Users of `cuda_fp16.h` and `cuda_bf16.h` headers are advised to disable host compilers strict aliasing rules based optimizations (e.g. pass `-fno-strict-aliasing` to host GCC compiler) as these may interfere with the type-punning idioms used in the `__half`, `__half2`, `__nv_bfloat16`, `__nv_bfloat162` types implementations and expose the user program to undefined behavior. Note, the headers suppress GCC diagnostics through: #pragma GCC diagnostic ignored `-Wstrict-aliasing`. This behavior may improve in future versions of the headers.

6.5.2. CUDA Math: Release 12.2

- **New Features**
  - CUDA Math APIs for `__half` and `__nv_bfloat16` types received usability improvements, including host side <emulated> support for many of the arithmetic operations and conversions.
  - `__half` and `__nv_bfloat16` types have implicit conversions to/from integral types, which are now available with host compilers by default. These may cause build issues due to ambiguous overloads resolution. Users are advised to update their code to select proper overloads. To opt-out user may want to define the following macros (these macros will be removed in the future CUDA release):
    - `__CUDA_FP16_DISABLE_IMPLICIT_INTEGER_CONVERTS_FOR_HOST_COMPILERS__`
    - `__CUDA_BF16_DISABLE_IMPLICIT_INTEGER_CONVERTS_FOR_HOST_COMPILERS__`
- **Resolved Issues**
  - During ongoing testing, NVIDIA identified that due to an algorithm error the results of 64-bit floating-point division in default round-to-nearest-even mode could produce spurious overflow to infinity. NVIDIA recommends that all developers requiring strict IEEE754 compliance update to CUDA Toolkit 12.2 or newer. The affected algorithm was present in both offline compilation as well as just-in-time (JIT) compilation. As JIT compilation is handled by the driver, NVIDIA recommends updating to driver version greater than or equal to R535 (R536 on Windows) when IEEE754 compliance is required and when using JIT. This is a software algorithm fix and is not tied to specific hardware.
  - Updated the observed worst case error bounds for single precision intrinsic functions `__expf()`, `__exp10f()` and double precision functions `asinh()`, `acosh()`.

6.5.3. CUDA Math: Release 12.1

- **New Features**
  - Performance and accuracy improvements in `atanf`, `acosf`, `asinf`, `sinpif`, `cospif`, `powf`, `erff`, and `tgammaf`.

6.5.4. CUDA Math: Release 12.0

- **New Features**
  - Introduced new integer/fp16/bf16 CUDA Math APIs to help expose performance benefits of new DPX instructions. Refer to https://docs.nvidia.com/cuda/cuda-math-api/index.html.
- **Known Issues**
  - Double precision inputs that cause the double precision division algorithm in the default 'round to nearest even mode' produce spurious overflow: an infinite result is delivered where `DBL_MAX 0x7FEF_FFFF_FFFF_FFFF` is expected. Affected CUDA Math APIs: `__ddiv_rn()`. Affected CUDA language operation: double precision / operation in the device code.
- **Deprecations**
  - All previously deprecated undocumented APIs are removed from CUDA 12.0.

6.6. NVIDIA Performance Primitives (NPP)

6.6.1. NPP: Release 12.0

- **Deprecations**
  - Deprecating non-CTX API support from next release.
- **Resolved Issues**
  - A performance issue with the NPP `ResizeSqrPixel` API is now fixed and shows improved performance.

6.7. nvJPEG Library

6.7.1. nvJPEG: Release 12.2

- **New Features**
  - Added support for JPEG Lossless decode (process 14, FO prediction).
  - nvJPEG is now supported on L4T.

6.7.2. nvJPEG: Release 12.0

- **New Features**
  - Immproved the GPU Memory optimisation for the nvJPEG codec.
- **Resolved Issues**
  - An issue that causes runtime failures when nvJPEGDecMultipleInstances was tested with a large number of threads is resolved.
  - An issue with CMYK four component color conversion is now resolved.
- **Known Issues**
  - Backend NVJPEG_BACKEND_GPU_HYBRID - Unable to handle bistreams with extra scans lengths.
- **Deprecations**
  - The reuse of Huffman table in Encoder (nvjpegEncoderParamsCopyHuffmanTables).

Chapter 7. Notices

7.1. 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 acknowledgement, 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.

No license, either expressed or implied, is granted under any NVIDIA patent right, copyright, or other NVIDIA intellectual property right under this document. Information published by NVIDIA regarding third-party products or services does not constitute a license from NVIDIA to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property rights of the third party, or a license from NVIDIA under the patents or other intellectual property rights of NVIDIA.

Reproduction of information in this document is permissible only if approved in advance by NVIDIA in writing, reproduced without alteration and in full compliance with all applicable export laws and regulations, and accompanied by all associated conditions, limitations, and notices.

THIS DOCUMENT AND ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL NVIDIA BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA's aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms of Sale for the product.

7.2. OpenCL

OpenCL is a trademark of Apple Inc. used under license to the Khronos Group Inc.

7.3. Trademarks

NVIDIA and the NVIDIA logo are trademarks or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright