NVIDIA CUDA Toolkit

Release Notes for CUDA 11.3.1

RN-06722-001_v11.3 | May 2021

Table of Contents

List of Tables

Chapter 1. CUDA 11.3 Release Notes

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

1.1. CUDA Toolkit Major Component Versions

CUDA Components

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

For CUDA 11.3, the table below indicates the versions:

Table 1. CUDA 11.3 Component Versions

Component Name	Version Information	Supported Architectures
CUDA Runtime (cudart)	11.3.109	x86_64, POWER, Arm64
cuobjdump	11.3.58	x86_64, POWER, Arm64
CUPTI	11.3.111	x86_64, POWER, Arm64
CUDA cuxxfilt (demangler)	11.3.58	x86_64, POWER, Arm64
CUDA Demo Suite	11.3.58	x86_64
CUDA GDB	11.3.109	x86_64, POWER, Arm64
CUDA Memcheck	11.3.109	x86_64, POWER
CUDA NVCC	11.3.109	x86_64, POWER, Arm64
CUDA nvdisasm	11.3.58	x86_64, POWER, Arm64
CUDA NVML Headers	11.3.58	x86_64, POWER, Arm64
CUDA nvprof	11.3.111	x86_64, POWER, Arm64
CUDA nvprune	11.3.58	x86_64, POWER, Arm64
CUDA NVRTC	11.3.109	x86_64, POWER, Arm64
CUDA NVTX	11.3.109	x86_64, POWER, Arm64
CUDA NVVP	11.3.111	x86_64, POWER
CUDA Samples	11.3.58	x86_64, POWER, Arm64

Component Name	Version Information	Supported Architectures
CUDA Compute Sanitizer API	11.3.111	x86_64, POWER, Arm64
CUDA cuBLAS	11.5.1.109	x86_64, POWER, Arm64
CUDA cuFFT	10.4.2.109	x86_64, POWER, Arm64
CUDA cuRAND	10.2.4.109	x86_64, POWER, Arm64
CUDA cuSOLVER	11.1.2.109	x86_64, POWER, Arm64
CUDA cuSPARSE	11.6.0.109	x86_64, POWER, Arm64
CUDA NPP	11.3.3.95	x86_64, POWER, Arm64
CUDA nvJPEG	11.5.0.109	x86_64, POWER, Arm64
Nsight Compute	2021.1.1.5	x86_64, POWER, Arm64 (CLI only)
Nsight Windows NVTX	1.21018621	x86_64, POWER, Arm64
Nsight Systems	2021.1.3.14	x86_64, POWER, Arm64 (CLI only)
Nsight Visual Studio Edition (VSE)	2021.1.1.21111	x86_64 (Windows)
NVIDIA Linux Driver	465.19.01	x86_64, POWER, Arm64
NVIDIA Windows Driver	465.89	x86_64 (Windows)

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-runtime-and-driver-api-version.

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 enhanced compatibility is shown below. CUDA Enhanced Compatibility is described in detail in https://docs.nvidia.com/deploy/cuda-compatibility/index.html

Table 2. CUDA Toolkit and Minimum Required Driver Version for CUDA Enhanced Compatibility

CUDA Toolkit	Minimum Required Driver Version for CUDA Enhanced Compatibility	
	Linux x86_64 Driver Version	Windows x86_64 Driver Version
CUDA 11.0 to CUDA 11.3.1 Update 1	>=450.80.02	>=456.38

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 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
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

* 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.

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 http://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 http://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

1.2. What's New in CUDA 11.3.1

This section summarizes the changes in CUDA 11.3.1 (11.3 Update 1) since the 11.3.0 release.

General CUDA

- Support for Ubuntu 16.04 is deprecated. CUDA will drop support for Ubuntu 16.04 in CUDA 11.4

cuBLAS

- New Features:
  - Some new kernels have been added for improved performance but have the limitation that only host pointers are supported for scalars (for example, alpha and beta parameters). This limitation is expected to be resolved in a future release.
  - New epilogues have been added to support fusion in ML training. This includes:
    - ReLuBias and GeluBias epilogues that produce an auxiliary output which is used on backward propagation to compute the corresponding gradients.
    - DReLuBGrad and DGeluBGrad epilogues that compute the backpropagation of the corresponding activation function on matrix C, and produce bias gradient as a separate output. These epilogues require auxiliary input mentioned in the bullet above.
- Deprecations:
  - Linking with static cublas and cublasLt libraries on Linux now requires using gcc-5.2 and compatible or higher due to C++11 requirements in these libraries.
- Known Issues:

- To be able to access the fastest possible kernels through `cublasLtMatmulAlgoGetHeuristic()` you need to set CUBLASLT_MATMUL_PREF_POINTER_MODE_MASK in search preferences to CUBLASLT_POINTER_MODE_MASK_HOST or CUBLASLT_POINTER_MODE_MASK_NO_FILTERING. By default, heuristics query assumes the pointer mode may change later and only returns algo configurations that support both _HOST and _DEVICE modes. Without this, newly added kernels will be excluded and it will likely lead to a performance penalty on some problem sizes.

cuSPARSE

- New Features:
  - Introduced a new routine for sparse matrix - sparse matrix multiplication (`cusparseSpGEMMreuse`) where the output matrix structure is reused for multiple computation. The new routine supports CSR storage format and mixed-precision computation.
  - Sparse triangular solver adds support for COO format.
  - Introduced a new routine for sparse triangular solver with multiple right-hand sides `cusparseSpSM()`.
  - `cusparseDenseToSparse()` routine adds the conversion from dense matrix (row-major/column-major) to Blocked-ELL format.
  - Blocke-ELL format now support empty blocks
  - Better performance for Blocked-ELL SpMM with block size > 64, double data type, and alignments smaller than 128-byte on NVIDIA Ampere sm80.
  - All cuSPARSE APIs are now asynchronous on platforms that support stream ordered memory allocators https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#stream-ordered-querying-memory-support.
  - Improved NTVX trace with distinction between light calls and kernel routines
- Resolved Issues:
  - `cusparseCnnz_compress` produced wrong results when the number of rows are greater than 128 * resident CTAs.
  - `cusparseSnnz` produced wrong results for some particular sparsity pattern.
- Deprecations:
  - `cusparseXcsrsm2_zeroPivot`, `cusparseXcsrsm2_solve`, `cusparseXcsrsm2_analysis`, and `cusparseScsrsm2_bufferSizeExt` have been deprecated in favor of `cusparseSpSM` Generic APIs

1.3. General CUDA

- Stream ordered memory allocator enhancements. See https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#stream-ordered-memory-allocator for more information.
- CUDA Graph Enhancements:

- Enhancements to make stream capture more flexible: Functionality to provide read-write access to the graph and the dependency information of a capturing stream, while the capture is in progress. See `cudaStreamGetCaptureInfo_v2()` and `cudaStreamUpdateCaptureDependencies()`.
- User object lifetime assistance: Functionality to assist user code in lifetime management for user-allocated resources referenced in graphs. Useful when graphs and their derivatives and asynchronous executions have an unknown/unbounded lifetime not under control of the code that created the resource, such as libraries under stream capture. See `cudaUserObjectCreate()` and `cudaGraphRetainUserObject()`, or http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#cuda-user-objects.
- Graph Debug: New API to produce a DOT graph output from a given CUDA Graph.

- New Stream Priorities
  - The CUDA Driver API `cuCtxGetStreamPriorityRange()` now exposes a total of 6 stream priorities, up from the 3 exposed in prior releases.
- Expose driver symbols in runtime API
  - New CUDA Driver API `cuGetProcAddress()` and CUDA Runtime API `cudaDriverGetEntryPoint()` to query the memory addresses for CUDA Driver API functions.
- Support for virtual aliasing across kernel boundaries.
- Added support for Ubuntu 20.04.2 on x86_64 and Arm sbsa platforms.

1.4. CUDA Tools

1.4.1. CUDA Compilers

- Cu++flt demangler tool
- NVRTC versioning changes
- Preview support for `alloca()`.

1.4.2. Nsight Eclipse Plugin

- Eclipse versions 4.10 to 4.14 are currently supported in CUDA 11.3.

1.5. CUDA Libraries

1.5.1. cuFFT Library

- cuFFT shared libraries are now linked statically against libstdc++ on Linux platforms.

- Improved performance of certain sizes (multiples of large powers of 3, powers of 11) in SM86.

1.5.2. cuSPARSE Library

- Added new routine `cusparesSpSV` for sparse triangular solver with better performance. The new Generic API supports:
  - CSR storage format
  - Non-transpose, transpose, and transpose-conjugate operations
  - Upper, lower fill mode
  - Unit, non-unit diagonal type
  - 32-bit and 64-bit indices
  - Uniform data type computation

1.5.3. NVIDIA Performance Primitives (NPP)

- Added `nppiDistanceTransformPBA` functions.

1.6. Deprecated Features

The following features are deprecated in the current release of the CUDA software. The features 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.

CUDA Libraries

- cuSPARSE: `cusparseScsrsv2_analysis`, `cusparseScsrsv2_solve`, `cusparseXcsrsv2_zeroPivot`, and `cusparseScsrsv2_bufferSize` have been deprecated in favor of `cusparseSpSV`.

Tools

- Nsight Eclipse Plugin: Docker support is deprecated in Eclipse 4.14 and earlier versions as of CUDA 11.3, and Docker support will be dropped for Eclipse 4.14 and earlier in a future CUDA Toolkit release.

1.7. Resolved Issues

1.7.1. General CUDA

- Historically, the CUDA driver has serialized most APIs operating on the same CUDA context between CPU threads. In CUDA 11.3, this has been relaxed for kernel launches

such that the driver serialization may be reduced when multiple CPU threads are launching CUDA kernels into distinct streams within the same context.

1.7.2. cuRAND Library

- Fixed inconsistency between random numbers generated by GPU and host generators when CURAND_ORDERING_PSEUDO_LEGACY ordering is selected for certain generator types.

1.7.3. CUDA Math API

- Previous releases of CUDA were potentially delivering incorrect results in some Linux distributions for the following host Math APIs: `sinpi`, `cospi`, `sincospi`, `sinpif`, `cospif`, `sincospif`. If passed huge inputs like 7.3748776e+15 or 8258177.5 the results were not equal to 0 or 1. These have been corrected with this release.

1.8. Known Issues

1.8.1. cuBLAS Library

- The planar complex matrix descriptor for batched matmul has inconsistent interpretation of batch offset.
- Mixed precision operations with reduction scheme CUBLASLT_REDUCTION_SCHEME_OUTPUT_TYPE (might be automatically selected based on problem size by `cublasSgemmEx()` or `cublasGemmEx()` too, unless `CUBLAS_MATH_DISALLOW_REDUCED_PRECISION_REDUCTION` math mode bit is set) not only stores intermediate results in output type but also accumulates them internally in the same precision, which may result in lower than expected accuracy. Please use `CUBLASLT_MATMUL_PREF_REDUCTION_SCHEME_MASK` or `CUBLAS_MATH_DISALLOW_REDUCED_PRECISION_REDUCTION` if this results in numerical precision issues in your application.

1.8.2. cuFFT Library

- cuFFT planning and plan estimation functions may not restore correct context affecting CUDA driver API applications.
- Plans with strides, primes larger than 127 in FFT size decomposition and total size of transform including strides bigger than 32GB produce incorrect results.

1.8.3. cuSOLVER Library

- For values N<=16, `cusolverDn[S|D|C|Z]syevjBatched` hits out-of-bound access and may deliver the wrong result. The workaround is to pad the matrix A with a diagonal matrix D such that the dimension of [A 0 ; 0 D] is bigger than 16. The diagonal entry D(j,j) must be

bigger than maximum eigenvalue of A, for example, norm(A, 'fro'). After the syevj, W(0:n-1) contains the eigenvalues and A(0:n-1,0:n-1) contains the eigenvectors.

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.

VESA DisplayPort

DisplayPort and DisplayPort Compliance Logo, DisplayPort Compliance Logo for Dual-mode Sources, and DisplayPort Compliance Logo for Active Cables are trademarks owned by the Video Electronics Standards Association in the United States and other countries.

HDMI

HDMI, the HDMI logo, and High-Definition Multimedia Interface are trademarks or registered trademarks of HDMI Licensing LLC.

OpenCL

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

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

NVIDIA Corporation | 2788 San Tomas Expressway, Santa Clara, CA 95051
www.nvidia.com