NVIDIA CUDA TOOLKIT 10.2.89

RN-06722-001 _v10.2 | November 2020

Release Notes for Windows, Linux, and Mac OS

TABLE OF CONTENTS

LIST OF TABLES

Chapter 1. CUDA TOOLKIT MAJOR COMPONENTS

This section provides an overview of the major components of the CUDA Toolkit and points to their locations after installation.

Compiler

The CUDA-C and CUDA-C++ compiler, **`nvcc`**, is found in the **`bin/`** directory. It is built on top of the NVVM optimizer, which is itself built on top of the LLVM compiler infrastructure. Developers who want to target NVVM directly can do so using the Compiler SDK, which is available in the **`nvvm/`** directory.

Please note that the following files are compiler-internal and subject to change without any prior notice.

- any file in **`include/crt`** and **`bin/crt`**
- **`include/common_functions.h`**, **`include/device_double_functions.h`**, **`include/device_functions.h`**, **`include/host_config.h`**, **`include/host_defines.h`**, and **`include/math_functions.h`**
- **`nvvm/bin/cicc`**
- **`bin/cudafe++`**, **`bin/bin2c`**, and **`bin/fatbinary`**

Tools

The following development tools are available in the **`bin/`** directory (except for Nsight Visual Studio Edition (VSE) which is installed as a plug-in to Microsoft Visual Studio, Nsight Compute and Nsight Systems are available in a separate directory).

- IDEs: **`nsight`** (Linux, Mac), Nsight VSE (Windows)
- Debuggers: **`cuda-memcheck`**, **`cuda-gdb`** (Linux), Nsight VSE (Windows)
- Profilers: Nsight Systems, Nsight Compute, **`nvprof`**, **`nvvp`**, Nsight VSE (Windows)
- Utilities: **`cuobjdump`**, **`nvdisasm`**

Libraries

The scientific and utility libraries listed below are available in the **`lib/`** directory (DLLs on Windows are in **`bin/`**), and their interfaces are available in the **`include/`** directory.

- **`cublas`** (BLAS)
- **`cublas_device`** (BLAS Kernel Interface)

- **`cuda_occupancy`** (Kernel Occupancy Calculation [header file implementation])
- **`cudadevrt`** (CUDA Device Runtime)
- **`cudart`** (CUDA Runtime)
- **`cufft`** (Fast Fourier Transform [FFT])
- **`cupti`** (CUDA Profiling Tools Interface)
- **`curand`** (Random Number Generation)
- **`cusolver`** (Dense and Sparse Direct Linear Solvers and Eigen Solvers)
- **`cusparse`** (Sparse Matrix)
- **`libcu++`** (CUDA Standard C++ Library)
- **`nvJPEG`** (JPEG encoding/decoding)
- **`npp`** (NVIDIA Performance Primitives [image and signal processing])
- **`nvblas`** ("Drop-in" BLAS)
- **`nvcuvid`** (CUDA Video Decoder [Windows, Linux])
- **`nvgraph`** (CUDA nvGRAPH [accelerated graph analytics])
- **`nvml`** (NVIDIA Management Library)
- **`nvrtc`** (CUDA Runtime Compilation)
- **`nvtx`** (NVIDIA Tools Extension)
- **`thrust`** (Parallel Algorithm Library [header file implementation])

CUDA Samples

Code samples that illustrate how to use various CUDA and library APIs are available in the **`samples/`** directory on Linux and Mac, and are installed to **`C:\ProgramData\NVIDIA Corporation\CUDA Samples`** on Windows. On Linux and Mac, the **`samples/`** directory is read-only and the samples must be copied to another location if they are to be modified. Further instructions can be found in the *Getting Started Guides* for Linux and Mac.

Documentation

The most current version of these release notes can be found online at http://docs.nvidia.com/cuda/cuda-toolkit-release-notes/index.html. Also, the **`version.txt`** file in the root directory of the toolkit will contain the version and build number of the installed toolkit.

Documentation can be found in PDF form in the **`doc/pdf/`** directory, or in HTML form at **`doc/html/index.html`** and online at http://docs.nvidia.com/cuda/index.html.

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 1. 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.

Table 1 CUDA Toolkit and Compatible Driver Versions

CUDA Toolkit	Linux x86_64 Driver Version	Windows x86_64 Driver Version
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

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

CUDA-GDB Sources

CUDA-GDB sources are available as follows:

- For CUDA Toolkit 7.0 and newer, in the installation directory **`extras/`**. The directory is created by default during the toolkit installation unless the **`.rpm`** or **`.deb`** package installer is used. In this case, the **`cuda-gdb-src`** package must be manually installed.
- For CUDA Toolkit 6.5, 6.0, and 5.5, at https://github.com/NVIDIA/cuda-gdb.
- For CUDA Toolkit 5.0 and earlier, at ftp://download.nvidia.com/CUDAOpen64/.

- Upon request by sending an e-mail to mailto:oss-requests@nvidia.com.

Chapter 2. CUDA 10.2 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.

This release, cuBLAS patch build 10.2.2, contains fixes for several cuBLAS issues. See CUDA Libraries.

2.1. General CUDA

- Added support for CUDA Virtual Memory Management APIs.
- 10.2 now includes **`libcu++`**, a parallel standard C++ library for GPUs.
- The following new operating systems are supported by CUDA. See the System Requirements section in the NVIDIA CUDA Installation Guide for Linux for a full list of supported operating systems. Note that support for RHEL 6.x is deprecated and support will be dropped in the next release of CUDA.
  - Fedora 29
  - Red Hat Enterprise Linux (RHEL) 7.x and 8.x
  - OpenSUSE 15.x
  - SUSE SLES 12.4 and SLES 15.x
  - Ubuntu 16.04.6 LTS and Ubuntu 18.04.3 LTS
- CUDA 10.2 (Toolkit and NVIDIA driver) is the last release to support macOS for developing and running CUDA applications. Support for macOS will not be available starting with the next release of CUDA.
- CUDA Graphs APIs now support updates to node parameters in instantiated graphs.
- CUDA 10.2 includes new interop APIs (**`NVSci*`** libraries for buffer allocation, synchronization, and streaming APIs). These are beta and the APIs may change in future CUDA releases.
- The 1D linear texture size limit supported for Maxwell+ (i.e. GM20x+) GPUs in CUDA is now 2^28 (up from 2^27).

2.2. CUDA Tools

2.2.1. CUDA Compilers

- The following compilers are supported as host compilers in **`nvcc`**:
  - Clang 8.0
  - Xcode 10.2
- Added **`--suppress-debug-info`** option in PTXAS.
- Added **`--compile-as-tools-patch`** option to nvcc to support the new CUDA sanitizer tool that allows users to insert custom patches in the code. In addition, nvcc now documents the existing flag **`--keep-device-functions`**. This flag can be used to preserve external linkage **`__device__`** functions in the generated PTX, when using whole program compilation mode.
- The following PTX instructions are supported in inline assembly:
  - New shapes for mma.sync
  - **`m16n8k8`** for HMMA
  - **`m8n8k16`** and **`m8n8k32`** for IMMA
  - **`LDSM`** (variant supported is **`LDSM.16.{M88,MT88}.{1, 2,4}`**)
  - **`I2IP`**
  - **`Abs.f16, abs.f16x2`**
  - **`HSET.BM`**(**`set .f16/f16x2`** with integer destination)

  For more information on these instructions, refer to the PTX ISA documentation.
- Add option **`-forward-unknown-to-host-compiler`** to nvcc that allows forwarding of options not recognized by nvcc to the host compiler.
- nvcc now supports a new flag **`-Werror=all-warnings`** to turn all generated warnings into errors instead.
- **`nvcc`** now supports the "-MD" and "-MMD" flags related to dependency generation. For more information on the description of the new flags refer to the output of **`nvcc --help`** or refer to the NVCC documentation online.

2.2.2. CUDA Profiler

- For new features in Visual Profiler and **`nvprof`**, see the What's New section in the Profiler User's Guide.
- For new features available in CUPTI, see the What's New section in the CUPTI documentation.
- For system wide profiling, use Nsight Systems. Refer to the Nsight Systems Release Notes.
- For profiling specific CUDA kernels, use Nsight Compute. Refer to the Nsight Compute Release Notes.

2.2.3. CUDA-MEMCHECK

- For new features in CUDA-MEMCHECK, see the Release Notes in the CUDA-MEMCHECK documentation.

2.3. CUDA Libraries

This release of the CUDA toolkit is packaged with libraries that deliver new and extended functionality, bug fixes, and performance improvements for single and multi-GPU environments.

Also in this release the **`soname`** of the libraries has been modified to not include the minor toolkit version number. For example, the cuFFT library **`soname`** has changed from **`libcufft.so.10.1`** to **`libcufft.so.10`**. This is done to facilitate any future library updates that do not include API breaking changes without the need to relink.

2.3.1. cuBLAS Library

- Improved the performance on some large and other GEMM sizes (mostly M * N < 512*512, K >100) due to increased internal workspace size.

2.3.2. cuSOLVER Library

- **`cusolverMgGetrf`** and **`cusolverMgGetrs`** are added in cusolverMg library to support multiGPU LU.
- A new Tensor Cores Accelerated Iterative Refinement Solver (TCAIRS) is introduced. This is a linear solver to solve **`AX=B`** and is similar to the LAPACK **`Xgesv`** (or **`Xgetrf+Xgetrs`**) functions, but is different in that it uses reduced precision internally for acceleration and then refines the solution to achieve the corresponding accuracy of the solution. This solver support real and complex data types as well as single and multiple RHS systems of equations. Depending on the problem size and data types used, the observed speedup could reach over 5X. There are two types of APIs to access this solver:
  - The basic user friendly LAPACK style APIs:
    - **`cusolverDn<P1><P2>Dgesv_bufferSize`**
    - **`cusolverDn<P1><P2>Dgesv`**

    Where P1 is the final solution precision and P2 is the lowest precision used in the solver. For Example **`cusolverDnZKgesv_bufferSize`** will use tensor core accelerated half precision compute while the final solution will be double precision compute accurate.
- Added the following experimental expert APIs **`cusolverDnIRSXgesv`**, **`cusolverDnIRSXgesv_bufferSize`** and related APIs.

2.3.3. cuFFT Library

- Improved the performance and scalability for the following use cases:
  - multi-GPU non-power of 2 transforms
  - R2C and Z2D odd sized transforms
  - 2D transforms with small sizes and large batch counts

2.3.4. CUDA Math Library

- Added two absolute value APIs for half-precision **`__half`** and **`__half2`** data types: **`habs`**, **`habs2`**.
- Improved performance and accuracy in the following math functions: **`tanhf, round, roundf, erf, erff, sinf, cosf, sincosf, tanf, sinpif, cospif, sincospif, j0f, j1f, y0f, y1f`**

2.4. 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.

General CUDA

- Support for RHEL 6.x is deprecated with CUDA 10.1. It will be dropped in the next release of CUDA. Customers are encouraged to adopt RHEL 7.x to use new versions of CUDA.
- Microsoft Visual Studio versions 2011, 2012 and 2013 are now deprecated as host compilers for nvcc. Support for these compilers may be removed in a future release of CUDA.
- Support for the following compute capabilities are deprecated in CUDA 10.2. Note that support for these compute capabilities may be removed in a future release of CUDA.
  - sm_35 (Kepler)
  - sm_37 (Kepler)
  - sm_50 (Maxwell)

  Support for Kepler **`sm_30`** architecture based products will be dropped starting with the next release of CUDA.

  For more information on GPU products and compute capability, see this page
- For WMMA operations with floating point accumulators, the **`satf`** (saturate-to-finite value) mode parameter is deprecated. Using it can lead to unexpected results. See http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#wmma-description for details.

- Support for Linux cluster packages is deprecated and may be dropped in a future release of CUDA.
- CUDA 10.2 (Toolkit and NVIDIA driver) is the last release to support macOS for developing and running CUDA applications. Support for macOS will not be available starting with the next release of CUDA.

CUDA Libraries General

- The nvGRAPH library is deprecated. The library will no longer be shipped in future releases of the CUDA toolkit.
- Support for Kepler and Maxwell architectures (compute capability **`sm_35`** through **`sm_52`**) is deprecated.
- NPP Compression Primitives (JPEG Encode/Decode) are being deprecated and will be removed in the next release. Users of these functions are encouraged to use the nvJPEG library.

CUDA Libraries - NPP

- NPP Compression Primitives (JPEG Encode/Decode) are being deprecated and will be removed in the next release. Users of these functions are encouraged to use the nvJPEG library.

CUDA Libraries - nvJPEG

- Below APIs are deprecated in this release:
  - **`nvjpegStatus_t NVJPEGAPI nvjpegDecodePhaseOne`**
  - **`nvjpegStatus_t NVJPEGAPI nvjpegDecodePhaseTwo`**
  - **`nvjpegStatus_t NVJPEGAPI nvjpegDecodePhaseThree`**
  - **`nvjpegStatus_t NVJPEGAPI nvjpegDecodeBatchedPhaseOne`**
  - **`nvjpegStatus_t NVJPEGAPI nvjpegDecodeBatchedPhaseTwo`**
  - **`nvjpegStatus_t NVJPEGAPI nvjpegDecodeBatchedPhaseThree`**

  The functionality provided by these APIs will be offered by new functions in the next release.

CUDA Libraries - cuSOLVER

- The 32-bit API of cusolverMg multi-GPU library will be removed in the next release. Instead, a 64-bit API will be adopted for the following:
  - Symmetric eigensolver routines **`cusolverMgSyevd()`**
  - LU factorization and solver routines **`cusolverMgGetrf`** and **`cusolverMgGetrs`**.
- The expert interface **`cusolverDnIRSXgesv`** of the TCAIRS solver and its helper functions which are proposed as experimental expert APIs in this release will undergo minor changes. The basic user friendly API will remain the same. In summary:
  - The expert API will remove the input_data_type from its API since it is part of the structure Params.
  - The **`cusolverDnIRSInfosXXXX`** helpers function will no longer need the Params into their arguments.

- All instances of **`cudaDataType`** will be replaced by **`cusolverPrecType_t`**.

2.5. Resolved Issues

2.5.1. CUDA Compilers

- Fixed an issue where ptxas in some cases may optimize arithmetic shifts incorrectly.
- Fixed a crash during compilation when using **`-DCONSTEXPR=constexpr`** with the **`--expt-relaxed-constexpr`** option.
- Added documentation for the **`-time`** flag for nvcc. This flag can be used to measure the time take by nvcc and internal sub-components. See **`nvcc --help`** for details.
- Fixed an issue where ptxas crashes when assembling a PTX file with DWARF debug info generated by clang.

2.5.2. CUDA Libraries

The following issues have been resolved across the CUDA Libraries.
CUDA Libraries - NPP

- Fixed a race condition within NPP if user provided a custom created CUDA stream (default or non-blocking).
- Fixed an issue where incorrect values were returned by NPP Histogram helper functions. These values are used for defining the size of side buffers used by NPP Histogram main functions.
- NPP does not support non-blocking streams on Windows for devices working in WDDM mode.

CUDA Libraries - nvJPEG

- Fixed an issue with **`NVJPEG_BACKEND_HYBRID`** backend when restart markers are enabled.

CUDA Libraries - cuSOLVER

- Resolved a conflict of symbols between **`liblapack_static.a`** and **`libf2c`**.
- Resolve missing **`GKlib/string.o`** in **`libmetis_static.a`**

CUDA Libraries - cuBLAS

- This patch fixes a bug in cuBLAS that caused silent corruption of results on Volta and Turing architecture GPUs when the following three conditions were met:

  1. Batched GEMM APIs (`cublasGemmStridedBatchedEx()` or `cublasGemmBatchedEx()`) were called with a batch count above 65535.

  2. Mixed precision or fast math was turned on via the CUBLAS_TENSOR_OP_MATH math mode option (https://docs.nvidia.com/cuda/archive/10.2/cublas/index.html#cublassetmathmode).

3. The problem dimensions or pointer alignment did not allow cuBLAS to use tensor core accelerated kernels despite being requested and the fallback occurred to non-tensor core kernels (see https://docs.nvidia.com/cuda/archive/10.2/cublas/index.html#tensorop-restrictions for details).

- Resolved an issue where CUDA Graph capture with cuBLAS routines on multiple concurrent streams would have caused hangs or data corruption in some cases.
- Resolved an issue where strided batched GEMM routines can cause misaligned read errors.
- Resolved an issue where calls to **`cublas?gemm()`** with matrices A,B, or C having CUBLAS_OP_CONJ transposition flag, failed to report as not supported and similarly resulted in silent corruption of data.

CUDA Libraries - cuFFT

- Added missing documentation for the following functions:
  - **`cufftXtExecDescriptorR2C`**
  - **`cufftXtExecDescriptorC2R`**
  - **`cufftXtExecDescriptorZ2D`**
  - **`cufftXtExecDescriptorD2Z`**
- Resolved an issue where multi-GPU supported functionality omitted in-place restriction for all FFT plan types.
- Refer to this page for the temporary restriction on C2R/Z2D plans.

CUDA Libraries - cuRAND

- Starting with CUDA 10.0, the ordering of random numbers returned by **`MTGP32`** and **`MRG32k3a`** generators are no longer the same as previous releases despite being guaranteed by the documentation for the **`CURAND_ORDERING_PSEUDO_DEFAULT`** setting. This issue will be addressed in the next release by providing a new non-default option which returns the same ordering as previous releases and the default option will continue to provide the best performance option.

2.6. Known Issues

2.6.1. General CUDA

- No known issues at this time.

Acknowledgments

NVIDIA extends thanks to Professor Mike Giles of Oxford University for providing the initial code for the optimized version of the device implementation of the double-precision **`exp()`** function found in this release of the CUDA toolkit.

NVIDIA acknowledges Scott Gray for his work on small-tile GEMM kernels for Pascal. These kernels were originally developed for OpenAI and included since cuBLAS 8.0.61.2.