



# NVIDIA CUDA TOOLKIT 10.0.326

RN-06722-001 \_v10.0 | March 2019

**Release Notes for Windows, Linux, and Mac OS**



# TABLE OF CONTENTS

<b>Chapter 1. CUDA Toolkit Major Components.....</b>	<b>1</b>
<b>Chapter 2. Release Notes.....</b>	<b>4</b>
2.1. General CUDA.....	4
2.2. CUDA Tools.....	5
2.2.1. CUDA Compilers.....	5
2.2.2. CUDA Developer Tools.....	6
2.3. CUDA Libraries.....	6
2.3.1. nvJPEG.....	6
2.3.2. cuFFT Library.....	6
2.3.3. cuBLAS Library.....	7
2.3.4. NVIDIA Performance Primitives (NPP).....	7
2.3.5. cuSOLVER Library.....	7
2.4. Deprecated Features.....	7
2.5. Resolved Issues.....	8
2.5.1. General CUDA.....	8
2.5.2. CUDA Tools.....	9
2.5.3. CUDA Libraries.....	10
2.6. Known Issues.....	10
2.6.1. General CUDA.....	10
2.6.2. CUDA Samples.....	11
2.6.3. CUDA Tools.....	11
<b>Chapter 3. CUDA 10 Update 1 Release Notes.....</b>	<b>12</b>
<b>Chapter 4. Tegra SoC-based Product Release Notes.....</b>	<b>13</b>
4.1. New Features.....	13
4.2. Known Issues and Limitations.....	13
4.3. Resolved Issues.....	14

## LIST OF TABLES

Table 1	CUDA Toolkit and Compatible Driver Versions .....	2
---------	---	---



# 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).

- ▶ IDEs: **nsight** (Linux, Mac), Nsight VSE (Windows)
- ▶ Debuggers: **cuda-memcheck**, **cuda-gdb** (Linux), Nsight VSE (Windows)
- ▶ Profilers: **nvprof**, **nvvp**, Nsight VSE (Windows)
- ▶ Utilities: **cuobjdump**, **nvdiasm**, **gpu-library-advisor**

### 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)
- ▶ **cuda\_occupancy** (Kernel Occupancy Calculation [header file implementation])
- ▶ **cudadevrt** (CUDA Device Runtime)

- ▶ **cuda** (CUDA Runtime)
- ▶ **cufft** (Fast Fourier Transform [FFT])
- ▶ **cupti** (Profiling Tools Interface)
- ▶ **curand** (Random Number Generation)
- ▶ **cusolver** (Dense and Sparse Direct Linear Solvers and Eigen Solvers)
- ▶ **cuspars** (Sparse Matrix)
- ▶ **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. 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 10.0.130	>= 410.48	>= 411.31

CUDA Toolkit	Linux x86_64 Driver Version	Windows x86_64 Driver Version
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/CUDAAOpen64/>.
- ▶ Upon request by sending an e-mail to <mailto:oss-requests@nvidia.com>.

# Chapter 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>.

### 2.1. General CUDA

- ▶ CUDA 10.0 adds support for the Turing architecture (compute\_75 and sm\_75).
- ▶ CUDA 10.0 adds support for new programming constructs called CUDA Graphs, a new asynchronous task-graph programming model that enables more efficient launch and execution. See the API documentation for more information.
- ▶ [Warp matrix](#) functions now support additional matrix shapes 32x8x16 and 8x32x16. Warp matrix functions also include the ability (experimental in CUDA 10.0) to perform sub-byte operations (4-bit unsigned, 4-bit signed and 1-bit) using the Tensor Cores.
- ▶ Added support for CUDA-Vulkan and CUDA-DX12 interoperability APIs.
- ▶ Added support for a new instruction **nanosleep** that suspends a thread for a specified duration.
- ▶ Added 6.3 version of the Parallel Thread Execution instruction set architecture (ISA). For more details on new (sm\_75 target, wmma, nanosleep, FP16 atomics) and deprecated instructions, see this [section](#) in the PTX documentation.
- ▶ Starting with CUDA 10.0, the CUDA runtime is compatible with specific older NVIDIA drivers. A new package called “cuda-compat-<version>” is included in the toolkit installer packages. For more information on compatibility, see the [section](#) in the Best Practices Guide.
- ▶ 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.
  - ▶ Ubuntu 18.04 LTS\*
  - ▶ Ubuntu 14.04 LTS
  - ▶ SUSE SLES 15
  - ▶ OpenSUSE Leap 15
- ▶ Added support for peer-to-peer (P2P) with CUDA on Windows (WDDM 2.0+ only).



- ▶ Added a new CUDA sample to demonstrate multi-device cooperative group APIs.
- ▶ CUDA samples are now also available on GitHub: <https://github.com/NVIDIA/cuda-samples>.
- ▶ Added APIs to retrieve the LUID of CUDA devices (**cuDeviceGetLuid**).
- ▶ Added **cudaLimitMaxL2FetchGranularity** in the device management APIs (**cuDeviceGetLimit**) to set the maximum fetch granularity of L2 (in Bytes).
- ▶ The **cudaDeviceProp** struct now includes the device UUID.
- ▶ Added support for synchronization across multiple devices with Cooperative Groups (**cuLaunchCooperativeKernelMultiDevice**) on Windows in TCC mode.
- ▶ Added the ability to lock clocks in **nvidia-smi** and NVML (**nvmlDeviceSetGpuLockedClocks** and **nvmlDeviceResetGpuLockedClocks** APIs). The following commands can be used in **nvidia-smi**:
  - ▶ `$ nvidia-smi -lgc/--lock-gpu-clock <minGpuClock, maxGpuClock>`
  - ▶ `$ nvidia-smi -rgc/--reset-gpu-clock`

## 2.2. CUDA Tools

### 2.2.1. CUDA Compilers

- ▶ The following compilers are supported as host compilers in **nvcc**:
  - ▶ Clang 6.0
  - ▶ Microsoft Visual Studio 2017\* (RTW, Update 8 and later)
  - ▶ Xcode 9.4
  - ▶ XLC 16.1.x
  - ▶ ICC 18
  - ▶ PGI 18.x (with -std=c++14 mode)
  - ▶ *\*Starting with CUDA 10.0, **nvcc** supports all versions of Visual Studio 2017 (past and upcoming updates)*
- ▶ A new libNVVM API function, **nvvmLazyAddModuleToProgram**, is introduced. This function should be used for adding the **libdevice** module (and other similar modules) to a program to make it more efficient. Refer to the libNVVM specification for more information about this function and its use.
- ▶ **nvcc** has added the **--extensible-whole-program** (or **-ewp**) option. This can be used to do whole-program optimizations (as are done in default compiles), but allows calls to certain external functions, particularly the functions in **libcudadevrt**. With this option one could use cuda-device-parallelism features without using separate compilation. However, for this option a link must happen. This link happens automatically when **nvcc** is used to create the executable, but if you are using the host linker directly, then you will need to perform an explicit **nvcc -dlink** step.
- ▶ Warp matrix functions (wmma) were first introduced in PTX ISA version 6.0 as a preview feature, and are now fully supported retroactively from PTX ISA version 6.0 onwards.

## 2.2.2. CUDA Developer Tools

- ▶ CUDA 10.0 now includes Nsight Compute, a suite of new developer tools for profiling and debugging supported on Windows, Linux and Mac.
- ▶ For new features (support for Turing compute capability, profiling of OpenMP applications, CUDA graphs) in CUPTI, see the [Changelog](#) section in the CUPTI documentation.
- ▶ For new features in CUDA-MEMCHECK, see the [Release Notes](#) section in the CUDA-MEMCHECK documentation.
- ▶ OpenMP tools interface is now supported in **nvprof**.
- ▶ Profiler now supports version 3 of NVIDIA Tools Extension API (NVTX).

## 2.3. CUDA Libraries

This release of the toolkit includes optimized libraries for Turing architecture and performance-tuned across single and multi-GPU environments. Also this release introduces a new library, nvJPEG, for GPU accelerated hybrid JPEG decoding.

### 2.3.1. nvJPEG

- ▶ A new library for GPU-accelerated JPEG decoding. nvJPEG supports decoding of single and batched images, color space conversion, multiple phase decoding, and hybrid decoding using both CPU and GPU.
- ▶ Provides low-latency decoder for common JPEG formats used in computer vision applications such as image classification, object detection and image segmentation.
- ▶ nvJPEG can accelerate data loading and pre-processing for DL training with GPU-accelerated augmentation such as translation, zoom, scale, crop, flip, and others.
- ▶ nvJPEG can perform JPEG decoding and resizing in real-time for applications that demand low-latency deep learning inference.
- ▶ Key Features include:
  - ▶ Hybrid decoding using both the CPU and the GPU
  - ▶ Single image and batched image decoding
  - ▶ Color space conversion to RGB, BGR, RGBI, BGRI, and YUV, and
  - ▶ Single and multi phase decoding

### 2.3.2. cuFFT Library

- ▶ This release includes strong FFT performance scaling across 8 and 16-GPU systems.
- ▶ This release also enables applications with large FFT models to be used on dense GPU systems.
- ▶ Added support for R2C and C2R for multi-GPU, and improved performance of 1D/2D/3D R2C/C2R for both single and multi-GPU FFTs.

### 2.3.3. cuBLAS Library

- ▶ Includes Turing architecture-optimized mixed-precision GEMMs for Deep Learning applications.
- ▶ Added batched GEMV (General Matrix Vector Multiplication) support for mixed precision (FP16 input and output, and FP32 accumulation) to enable deep learning RNNs using attention models.
- ▶ Several improvements made to API logging, including logging the following formats:
  - ▶ Append print,
  - ▶ Print local time,
  - ▶ Append printing version,
  - ▶ Append synchronziation via mutex use.
  - ▶ Added different levels of logging where detailed information can be printed about gemm, such as:
    - ▶ Tensor-Core vs. Non-Tensor Core
    - ▶ Tile sizes and other performance options that are used internally, and
    - ▶ Grid dimensions and kernel name.

### 2.3.4. NVIDIA Performance Primitives (NPP)

- ▶ Added batched resize for 32F and U8 image formats.
- ▶ This release also extends the image resize functionality to batches of variable sized regions of interest (ROI).
- ▶ Improved performance for several image processing primitives.

### 2.3.5. cuSOLVER Library

- ▶ Improved performance of the following dense linear algebra routines:
  - ▶ Cholesky factorization,
  - ▶ Symmetric eigensolver,
  - ▶ Generalized symmetric eigensolver, and
  - ▶ QR factorization.

## 2.4. Deprecated Features

The following features are deprecated in the current release of the CUDA software. The features will 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 now deprecated starting with CUDA 10.0. It may be dropped in a future release of CUDA. Customers are encouraged to adopt RHEL 7.x to use new versions of CUDA.
- ▶ The usage of the following attributes of a pointer returned by `cudaPointerGetAttributes::memoryType` and `isManaged` is deprecated and will be removed in the next release of CUDA.
- ▶ The following compilers are no longer supported as host compilers for `nvcc`:
  - ▶ PGI 17.x
  - ▶ Microsoft Visual Studio 2010
  - ▶ Clang versions lower than 3.7
- ▶ Microsoft Visual Studio 2011 is now deprecated as a host compiler for `nvcc`. Support may be removed in a future release of CUDA.
- ▶ 32-bit tools are no longer supported starting with CUDA 10.0.
- ▶ Usage of non-sync versions of warp instructions (`shfl` and `vote`) is deprecated on Volta+ architectures.
- ▶ Installation of the CUDA Toolkit using the `.run` package no longer supports in-execution permission elevation to admin privileges.
- ▶ The following samples are no longer available in the CUDA toolkit:
  - ▶ `simpleDevLibCUBLAS`
  - ▶ `cdpLUdecomposition`

## CUDA Libraries

- ▶ The nvGRAPH library is deprecated. This library will no longer be shipped in the future releases of CUDA.
- ▶ The cuBLAS library, to support the ability to call the same cuBLAS APIs from within the device routines (`cublas_device`), is dropped starting with CUDA 10.0.

## 2.5. Resolved Issues

### 2.5.1. General CUDA

- ▶ Fixed a bug in `conjugateGradientMultiDeviceCG` sample where it would fail to run on some Volta GPUs.
- ▶ Fixed an issue with the GPU memory resource consumption of the MPS server on Volta V100.
- ▶ Fixed an issue that caused a 6% performance difference in FAHBench (DP\_average test) with OpenCL.
- ▶ Fixed a memory allocation issue in the CUDA runtime (`cudart`) library that was reported as an error by AddressSanitizer.
- ▶ Fixed an issue when NVML would not show all active NVLinks (via `nvidia-smi nvlink -s`) after a reboot of the system.

- ▶ Fixed an issue with the CUDA sample **reduction** (under 6\_Advanced/reduction), where the sample would fail when reducing less than 16384 elements.

## 2.5.2. CUDA Tools

- ▶ Fixed a bug in **nvprof** where **--print-nvlink-topology** would crash on POWER 8 systems.
- ▶ Fixed an issue with **nvprof** where system memory related metrics would fail to be collected on Pascal GPUs.
- ▶ Fixed an issue where **nvprof** would result in a crash when no kernel is specified in an **nvtx** range.
- ▶ Fixed an issue where the **--profile-child-processes** option with **nvprof** would result in no profile data being collected.
- ▶ Fixed a **NullPointerException** in some OpenMP codes when running the Visual Profiler.
- ▶ Fixed an issue with Nsight Eclipse Edition that was preventing auto-complete from working with CUDA.
- ▶ Fixed an issue in **nvvp** where the CPU profiling button should be disabled if tracing is enabled. This issue would result in a failure to generate timelines.
- ▶ Fixed an issue in **nvvp** to show the correct topology and link information for GPUs in a POWER 9 system.
- ▶ Fixed an issue where profiling of CUDA applications would result in invalid profiling results on Azure GPU instances.
- ▶ Fixed an issue in **ptxas** where the **vmax** instruction in specific cases (**vmax2.s32.s32.s32**) would result in incorrect results.
- ▶ Fixed an issue in **nvcc** where an error would be reported in some cases in **relaxed\_constexpr** mode for references to host variables in device code.
- ▶ Fixed an issue in the visual profiler where the CPU profiling does not default correctly for PGI.
- ▶ Fixed an issue where **cuda-gdb** would assert when running some OpenACC codes with **break\_on\_launch** set.
- ▶ Fixed an issue where **nvprof** was incorrectly calculating the percentage overhead for NVLink (see **nvlink\_overhead\_\* metrics**).
- ▶ Fixed an issue which prevented tracing and profiling (event and metric collection) for multidevice cooperative kernels, that is, kernels launched by using the API functions **cudaLaunchCooperativeKernelMultiDevice** or **cuLaunchCooperativeKernelMultiDevice**.
- ▶ Fixed an issue in **nvcc** where an internal compiler error is encountered when generating debug info for code containing a pointer to data member or member function.
- ▶ Fixed an issue where **nvcc** would return an error for a template that is defined inside an unnamed namespace.
- ▶ Fixed an issue in the CUDA compiler where using **\_\_shfl\_xor\_sync()** in some cases would cause kernels to hang on Volta GPUs.

## 2.5.3. CUDA Libraries

- ▶ Fixed the following performance issues with cuBLAS:
  - ▶ SGEMM (FP32) on V100/Ubuntu 16.04,
  - ▶ INT8 GEMM on P4 for small input m and n sizes, and
  - ▶ Batched CGEMM performance on Tesla V100.
- ▶ Fixed performance issue in cuRAND for Philox one-at-a-time random number generation.
- ▶ Improved cuBLAS performance and heuristics selection for large K matrices.
- ▶ Added METIS reordering in the cuSOLVER sample for sparse linear solver.
- ▶ Fixed an issue with cuSOLVER QR factorization (**geqrf**) that sometimes returned R factors containing a non numeric value.
- ▶ Fixed overflow and underflow issues in cuSOLVER householder reflection and givens rotations.
- ▶ Fixed an issue with cuBLAS SGEMM algorithm 102 that occasionally returned non-deterministic results.
- ▶ Fixed an issue in cuSOLVER **cusolverDnDsyevj** that returned incorrectly sorted eigenvalues.
- ▶ Improved dense SVD performance in cuSOLVER for tall skinny sizes by adding QR when  $m > n$ .
- ▶ Improved performance of dense QR with domino scheme in cuSOLVER.
- ▶ Fixed an issue in NPP **nppiCountInRange** routine that returned NPP\_RANGE\_ERROR when called from multiple threads.
- ▶ Fixed missing check for **uplo** parameter in cuSOLVER routines **cusolverDn<X>potrf** and **cusolverDn<X>potrf\_bufferSize** that resulted in incorrect return code.
- ▶ Fixed an issue with cuBLAS batched GEMM where it selected wrong kernels for bandwidth bound GEMMs.
- ▶ Fixed an issue in Thrust where **merge\_by\_key** uses the wrong element type in the default comparator.

## 2.6. Known Issues

### 2.6.1. General CUDA

- ▶ CUDA graphs with host nodes do not work under MPS with pre-Volta GPUs. Attempting to launch a graph in this configuration will result in **cuLaunchGraph()** potentially executing a portion of the graph and returning an error.
- ▶ On Red Hat Enterprise Linux and Fedora based systems, installation of the CUDA toolkit without installing the NVIDIA OpenGL libraries is not supported. See <https://docs.nvidia.com/cuda/cuda-installation-guide-linux/index.html#advanced-setup> for this option. This issue will be fixed in the next release of CUDA.
- ▶ Using the CUDA Graphs stream capture APIs with CUDA libraries (e.g. cuBLAS) may result in errors. This issue will be fixed in a patch update to CUDA 10.0.

- ▶ The nvJPEG library is currently only available on Linux x86-64 and is not supported on Linux POWER, Windows or Mac platforms. Support for these additional platforms may be added in a future release of CUDA.
- ▶ On **ppc64le** (Ubuntu 18.04 or RHEL 7.5) systems, the persistence daemon may not be started automatically after CUDA 10.0 is installed. The **nvidia-persistenced.service** (typically found under **/lib/systemd/system/**) file should be changed to as follows:

```
[Unit]
Description=NVIDIA Persistence Daemon
After=syslog.target

[Service]
Type=forking
PIDFile=/var/run/nvidia-persistenced/nvidia-persistenced.pid
Restart=always
ExecStart=/usr/sbin/nvidia-persistenced --verbose
ExecStopPost=/bin/rm -rf /var/run/nvidia-persistenced/*
TimeoutSec=300

[Install]
WantedBy=multi-user.target
```

Check the status of the persistence daemon with the following command:

```
$ systemctl status nvidia-persistenced
```

If it is not active, run the following command:

```
$ sudo systemctl enable nvidia-persistenced
```

## 2.6.2. CUDA Samples

- ▶ **cuda-memcheck** may report uninitialized reads in CUB's radix sort in the CUDA sample **smokeParticles**.
- ▶ Some graphic CUDA samples may terminate with an error when switching to the desktop using **Windows + D** key combination.

## 2.6.3. CUDA Tools

- ▶ For known issues in cuda-memcheck, see the [Known Issues](#) section in the cuda-memcheck documentation.
- ▶ Fixed a crash in **nvprune** when dealing with ELF objects with SHT\_NOBITS sections.

# Chapter 3.

## CUDA 10 UPDATE 1 RELEASE NOTES

CUDA 10.0 Update 1 is an update to the CUDA 10 Toolkit that includes new enhancements to CUDA libraries (cuBLAS).



# Chapter 4.

## TEGRA SOC-BASED PRODUCT RELEASE NOTES

The release notes for CUDA Tegra contain only information this is specific to the CUDA Tegra Driver and the mobile version of other CUDA components such as compilers, tools, libraries, and samples. The release notes for the desktop version of CUDA in the remaining chapters of this document also apply to CUDA Tegra. On Tegra, the CUDA Toolkit version is 10.

### 4.1. New Features

#### CUDA Tegra Driver

- ▶ Support for Drive AGX Pegasus added on QNX.
- ▶ Support for CUDA-Vulkan interoperability added on Nvidia Jetson and Android platforms.
- ▶ Support for Ubuntu 18.04 LTS added.
- ▶ CUDA kernel launch latency improved on QNX.
- ▶ Starting with CUDA 10.0, support for 14.04 LTS has been dropped.

#### CUDA Compiler

- ▶ Support for GCC 6.4 added.
- ▶ Support for Clang 5.0 & libc++ for Android added.

#### CUDA Samples

- ▶ A new Tegra sample for Unified Memory is added. This sample demonstrates significant speedup while using attached hints.

### 4.2. Known Issues and Limitations

#### CUDA Tegra Driver

- ▶ Support for 32-bit applications will no longer be available.

- ▶ Support for **cudaHostRegister** on QNX is not available, due to lack of compatible QNX kernels.
- ▶ The **cudaDeviceGetAttribute** methods returns false for the attribute **cudaDevAttrHostNativeAtomicSupported** for T194, because system wide atomics are currently not supported (for Nvidia Drive and Jetson platforms).
- ▶ Memory clock rate of dGPU is not reported correctly by **cudaGetDeviceProperties/cudaDeviceGetAttribute**.

## CUDA-GDB

- ▶ **Linux** : The **/dev/nvhost-dbg-gpu** and **/dev/nvhost-prof-gpu** nodes default to **0660** permissions for security reasons. It causes the following Dev Tools to be broken out of the box:
  - ▶ Tegra Graphics Debugger,
  - ▶ PerfWorks,
  - ▶ **nvprof**,
  - ▶ **cuda-memcheck**,
  - ▶ **cuda-gdb**.



Before using any of the Dev Tools listed above, please log into the board and run this command:

```
sudo chmod a+rw /dev/nvhost-dbg-gpu /dev/nvhost-prof-gpu.
```

- ▶ **Linux**: The **set cuda memcheck on** command in **cuda-gdb** does not have any effect.
- ▶ **QNX**: **ntoaarch64-gdb** and **cuda-qnx-gdb** may hang when executing the **runcommand**.

## NVPROF

- ▶ PC sampling is not supported.
- ▶ The Volta dGPU (GV100) is not supported.
- ▶ This release does not support HWPM context switching. That means that counters that are collected through the HWPM counter provider are available at the device level only at this time. This will be fixed in a future release.

## 4.3. Resolved Issues

### General CUDA

- ▶ During initialization, CUDA driver reserves a large chunk of CPU virtual address for its internal memory management. On QNX, this CPU virtual address reservation might take a considerable amount of time on systems with large physical memory. Because of this behavior, CUDA initialization might take more time on QNX Xavier as compared to earlier releases.
- ▶ CUDA allocators cannot make a single allocation greater than 4 GB on Tegra SoC memory. This applies to all allocations on Tegra iGPU and zero copy memory

allocations on Tegra dGPU. Application needs to make multiple allocations and aggregate them to create a large allocation.

## 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.

## Notice

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.

Information furnished is believed to be accurate and reliable. However, NVIDIA Corporation assumes no responsibility for the consequences of use of such information or for any infringement of patents or other rights of third parties that may result from its use. No license is granted by implication of otherwise under any patent rights of NVIDIA Corporation. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all other information previously supplied. NVIDIA Corporation products are not authorized as critical components in life support devices or systems without express written approval of NVIDIA Corporation.

## 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

© 2007-2019 NVIDIA Corporation. All rights reserved.  
[www.nvidia.com](http://www.nvidia.com)

