NVIDIA CUDA TOOLKIT 9.2.148

RN-06722-001 _v9.2 | August 2018

Release Notes for Windows, Linux, and Mac OS

TABLE OF CONTENTS

LIST OF TABLES

ERRATA

CUDA 9.2 (9.2.148) Update 1 is a bug fix update to the CUDA 9.2 Toolkit. The update includes fixes to issues in the CUDA Libraries (see Resolved Issues). The local installer packages for all supported operating systems include an updated NVIDIA driver. For more information on the driver versions, refer to Chapter 1.

New Features

CUDA Libraries

- cuBLAS 9.2.174 is an update to CUDA Toolkit 9.2 and is available separately as a patch update to CUDA 9.2. It includes the following changes:
  - Improved GEMM performance for large matrices of certain tile sizes.
  - Improved performance of half precision GEMM APIs with optimized algorithm selection heuristics for FP16 compute type.
- **cuBLAS:**
  - Improved performance of complex GEMM APIs for small n matrices.
  - Improved performance of half precision GEMM APIs - **`cublashgemm()`**, **`cublashgemmbatched()`** and **`cublashgemmstridedbatched()`**

Resolved Issues

CUDA Developer Tools

- Fixed an issue with invalid timestamps in the CUPTI records when using the activity APIs for serialized kernel tracing (via CUPTI activity kind CUPTI_ACTIVITY_KIND_KERNEL)
- Fixed an issue where cuda-gdb would assert when running some OpenACC codes with **`break_on_launch`** set.

CUDA Libraries

- **cuBLAS:**
  - The triangular linear system solver routine **`cublas<t>trsm()`** may produce incorrect results for some input combinations.

    - Fixed an issue with GEMM calls on V100 Tensor Core GPUs that caused incorrect results, in some cases. This affected all GEMM APIs used with the default algorithm **`CUBLAS_GEMM_DEFAULT_TENSOR_OP`**.
    - Fixed an issue with **`cublasGemmEx`**, which had non-deterministic and sometimes incorrect results when called with algorithm **`CUBLAS_GEMM_ALGO2_TENSOR_OP`**.
    - Fixed an issue with **`rotmg`** function that caused different results between cuBLAS and OpenBLAS for the same set of inputs.
    - Fixed an issue with **`cublasStrsm`** that resulted in different outputs across P100 and V100 systems for certain input matrix sizes.
    - Fixed an issue that generated empty PTX-JIT entries, resulting in a timeout when calling cuBLAS APIs on IBM POWER systems.
- **cuSPARSE:**
    - The sparse matrix-vector multiplication routine in based on the "mergepath" approach, **`cusparse<t>csrmv_mp()`** randomly crashes in cuSPARSE 9.2.

CUDA Compiler

- Fixed a regression in CUDA 9.2 **`nvcc`** related to function-try-blocks (see [except] in ISO C++ standard for the definition of function-try-blocks). In the presence of any function-try-blocks, compilation with **`nvcc`** used to abort with an assertion failure. Function-try-blocks could be replaced with try-blocks in functions to work around this issue unless function-try-blocks were used to catch and handle exceptions thrown by member initializers (see [class.base.init] in ISO C++ standard for the definition of member initializers). For example, compilation of the following code with **`nvcc`** would abort with an assertion failure:

  ```
  void f() try { /* do something */ } catch (…) { /* handle exception */ }
  ```

  To avoid the failure, the code could be rewritten as follows:

  ```
  void f() { try { /* do something */ } catch (…) { /* handle exception
   */ } }
  ```

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`**, **`nvdisasm`**, **`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)
- **`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** (Profiling Tools Interface)
- **curand** (Random Number Generation)
- **cusolver** (Dense and Sparse Direct Linear Solvers and Eigen Solvers)
- **cusparse** (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 7.0 (7.0.28)	>= 346.46	>= 347.62

CUDA Toolkit	Linux x86_64 Driver Version	Windows x86_64 Driver Version
CUDA 7.5 (7.5.16)	>= 352.31	>= 353.66
CUDA 8.0 (8.0.44)	>= 367.48	>= 369.30
CUDA 8.0 (8.0.61 GA2)	>= 375.26	>= 376.51
CUDA 9.0 (9.0.76)	>= 384.81	>= 385.54
CUDA 9.1 (9.1.85)	>= 390.46	>= 391.29
CUDA 9.2 (9.2.88)	>= 396.26	>= 397.44
CUDA 9.2 (9.2.148 Update 1)	>= 396.37	>= 398.26

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. NEW FEATURES

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

- Improved kernel launch latency (using the **`<<< >>>`** syntax and the **`cudaLaunchKernel API`**) for both multithreaded and multi-GPU code by up to a factor of 2 compared to CUDA 9.0.
- Added support for unified memory with address translation services (ATS) on IBM POWER9.
- Added arithmetic operators for the **`__half2`** data type and a volatile assignment operator for the **`__half`** data type.
- Added version 6.2 of the Parallel Thread Execution instruction set architecture (ISA). For details about new instructions (**`activemask`**, **`FP16`**, and **`atomics`**) and deprecated instructions, see Parallel Thread Execution ISA Version 6.2 in the PTX documentation.
- IPC functionality is now supported on Windows.
- Added P2P write and read bandwidth and latency metrics to the **`p2pBandwidthLatencyTest`** sample.
- Thrust now uses CUB v1.7.5.
- Added some performance optimizations in Thrust for the templated complex type.
- Added support for new operating systems. For a list of operating systems supported by CUDA, see the following information in the installation guides:
  - Windows system requirements
  - Mac OS X system requirements
  - Linux system requirements
- Changed **`CUDA_DEVICE_ORDER==FASTEST_FIRST`** to enumerate GPUs in descending order of performance.
- Added a new driver API **`cuStreamGetCtx`** to retrieve the context associated with a stream. This API is primarily used by the multidevice cooperative launch runtime API to ensure that the specified function's module is loaded in the right context.

- Added support for full core dump generation on Linux by using named pipes for MPS-based CUDA applications and CUDA applications that are not based on MPS.
- Added these new helper APIs for cooperative groups:
  - **`grid_dim()`** to get the 3-dimensional grid size
  - **`block_dim()`** to get the 3-dimensional block size

2.2. CUDA Tools

2.2.1. CUDA Compilers

- The following compilers are supported as host compilers in **`nvcc`**
  - Clang 5.0
  - GCC 7.*x*
  - Microsoft Visual Studio 2017 (RTW and Update 6)
  - PGI pgc++ 18.*x*
  - XLC 13.1.6
- **`__device__ / __constant__`** variables are now allowed to have an **`rvalue`** reference type.
- Functions in **`math_functions.hpp`** have been changed to use **`memcpy`** for type punning.
- Added support for **`std::tuple`**.
- Enabled **`pgcc`** to include some CUDA header files by defining CUDA-specific macros with GNU-style attributes.

2.2.2. CUDA Profiler

- Added new utilization and count metrics for Tensor Cores on GPUs based on the Volta architecture.
- Added CLI options for **`nvprof --trace`** <**`gpu`**,**`api`**> to show trace and profile information in the same output.
- Visual Profiler now includes a summary view to show the memory hierarchy.

2.2.3. CUDA Profiling Tools Interface (CUPTI)

For information about new features such as PCIe topology, new metrics, and new profiling scope in CUPTI, see Changelog in the CUPTI documentation.

- Added support in CUPTI to allow **`hwpm_active_warps`** and **`hwpm_active_cycles`** counters to be collected in a single pass.
- Added support for the NVTX v3 interface

2.2.4. CUDA-GDB

- CUDA now supports lightweight core dumps.

2.2.5. CUDA-MEMCHECK

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

2.3. CUDA Libraries

2.3.1. cuBLAS Library

- Improved performance for a range of small and large tile size matrices that are extensively used in RNN based speech and NLP models, Convolutional seq2seq (Fairseq) models, OpenAI research and other emerging DL applications. These sizes are optimized on the Tesla V100 architecture to deliver enhanced out-of-the-box performance.
- Added GEMM API logging for developers to trace the algorithm and dataset used during the last BLAS API call.
- Improved GEMM performance on Tesla V100 for single and double precision inputs.

2.3.2. NVIDIA Performance Primitives (NPP)

- Added support for NV12-to-RGB format conversion, which is important for deep learning because the decoder output format is NV12 and the typical input format for networks is RGB.
- Added primitives to convert real-valued images to complex-valued images and vice versa, for single-channeled images.
- Added a new NPP sample under CUDA samples called **`boundSegmentsNPP`**.

2.3.3. cuFFT Library

- Improved the performance for prime factor FFT sizes with fused bluestein kernels.
- A new memory-usage API provides an optional minimal work area policy setting that allows:
  - Transforms of type C2C to be supported with sizes of up to 4096 in any dimension
  - Transforms of type Z2Z to be supported with sizes of up to 2048 in any dimension
- Provided a new static library that supports only standard cuFFT APIs, that is, without the callbacks. Supporting standard only cuFFT APIs removes the dependency on the CUDA compiler and callback functionality for certain deployments.

2.3.4. cuSOLVER Library

- Added the following sparse matrix reordering options:

- A zero-free diagonal reordering option to permute rows of a sparse matrix such that there are no zeroes on diagonals after reordering.
- An option for matrix reordering by using the METIS library. This option typically delivers smaller zero fill-in than nested dissection during factorization.

2.3.5. cuSPARSE Library

- Significantly improved the performance of the merge-path-based sparse matrix-vector multiplication routines (**`csrmv_mp`** and **`csrmvEx`**) .
- Added a new triangular solver (**`csrsm2`**) that provides the same functionality as the existing **`csrsv2`** but extends support for multiple right-hand-side vectors.
- Added a batched pentadiagonal solver that supports 5-vector matrices and interleaved data layouts. This solver is intended for large batches (thousands) of small matrices (size in the hundreds).

2.3.6. Thrust Library

- CUB 1.7.5 has been integrated as a device back end for Thrust.

Chapter 3. 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

- The execution control APIs in CUDA will be removed in the next release of CUDA and will no longer be available. These APIs are as follows:
  - **`cudaConfigureCall`**
  - **`cudaLaunch`**
  - **`cudaSetupArgument`**
- The NVIDIA Video Decoder (NVCUVID) is deprecated. Instead, use the NVIDIA Video Codec SDK. As of CUDA 9.2, the following files are still available under the CUDA installation directory (for example, for Linux, this directory may be **`/usr/local/cuda/include`**). These files will be removed in the next release of the CUDA Toolkit:
  - **`dynlink_cuviddec.h`**
  - **`dynlink_nvcuvid.h`**
  - **`dynlink_cuda.h`**
  - **`dynlink_cuda_cuda.h`**
  - Windows **`nvcuvid`** static library: **`\lib\x64\nvcuvid.lib`**
- In nvcc, the **`--use-local-env`** option no longer requires **`--cl-version`** and **`--cl-version`** is now ignored. With this change, **`nvcc`** detects the Microsoft Visual Studio compiler version from the local environment without relying on **`--cl-version`**.
- 32-bit application developer tools for Windows is deprecated and will no longer be supported in the next release of CUDA. As of CUDA 9.2, 32-bit support for the following tools on Windows (and associated files) is still available but will be removed in the next release:
  - Nsight Visual Studio Edition (CUDA debugger, CUDA trace/timeline, CUDA profiler)

  - Nvprof and Visual Profiler (CUPTI)
    - **`lib\Win32\cuinj32_xx.dll`**
    - **`extras\CUPTI\libWin32\cupti.lib, extras\CUPTI \libWin32\cupti32_xx.dll`**
  - Cuda-memcheck
- In the next release of CUDA, Microsoft Visual Studio 2010 will no longer be supported.
- Starting with R396, the OpenCL ICD loader version will be reported as 2.2 instead of 2.0. Note that there is no change in the OpenCL version (1.2) supported by NVIDIA.
- Starting with R396, the Fermi architecture (**`sm_2x`**)
- The Runfile installation method includes a feature to elevate permissions during the installation. This feature is deprecated as of CUDA 9.2, and will be removed at a later release. Check the Linux installation guide for more details.

CUDA Libraries

- Since CUDA 5.0, the cuBLAS library has supported the ability to call the same cuBLAS APIs from within device routines, i.e. kernels. These routines are implemented using the Dynamic Parallelism feature, which is available starting with the Kepler generation of GPUs.The device library (**`cublas_device`**) that enables this feature, is deprecated in this release and will be dropped starting next release. NOTE: none of the main cuBLAS library functionality and the APIs that can be called from the host, is impacted.

Chapter 4. RESOLVED ISSUES

4.1. General CUDA

- Fixed incorrect memory access issues when **oceanFFT** is running on GPUs based on the Volta architecture.
- The macros in cooperative groups **cg_assert()** and **die()** have been renamed to **_CG_ASSERT()** and **_CG_ABORT()**
- Fixed a crash with the **simpleIPC** CUDA sample on 16-GPU systems.
- Fixed an issue in NVML to allow users to set application clocks by using **nvidia-smi** (**nvidia-smi -ac**) without requiring root privileges on GPUs based on the Pascal and later architectures.
- Improved the performance of the PTX JIT cache in a multiprocess environment. See the CUDA documentation about JIT cache management for more information.
- Fixed a bug in the CUDA runtime that caused a **pthread_mutex** hang on the POWER platform when running some CUDA applications.
- Fixed a bug in the CUDA memory allocator when using **cudaDeviceSetLimit()** that could result in heap corruption.
- Fixed a bug in the **shfl_scan** CUDA sample code when converting unsigned **int** to **uchar4**.
- In R396, removed **nv_flush_caches()** for recent kernels (2.6.25 and greater), which implement cache flush in **pageattr.c**.
- Fixed a bug where the CUDA samples would not load when multiple versions of Microsoft Visual Studio are installed on the system.
- In R396, fixed **nvmlDeviceGetTopologyCommonAncestor** to return **NVML_ERROR_NOT_SUPPORTED** instead of **NVML_ERROR_UNKNOWN** for GPUs that do not support this API.

4.2. CUDA Tools

4.2.1. CUDA Compilers

- Fixed an issue in the CUDA compiler, where in some cases, mixing **`shfl`** and certain carry operations on **`sm_70`** produces incorrect code.
- Fixed an issue in the CUDA compiler with incorrect constant folding in the presence of **`mul.wide.u16`** instructions.
- Fixed a crash in PTXAS compiling certain PTX files that contain debugging information.
- Fixed an incompatibility issue with **`nvcc`** and **`glibc`** 2.26
- In some cases, when NVVM IR is compiled with **`libNVVM`** on GCC with debugging information (**`-g`**), PTXAS may fail with the following message:
  Parsing error near '-': syntax error
- Fixed a crash in PTXAS when a user-defined label is present at the start of a function.
- Fixed a performance issue by tuning the CUDA compiler's heuristics for application code that may contain a large number of switch statements.
- Fixed an issue in the CUDA compiler to significantly reduce the compilation time for certain kernels that include complicated array element access patterns.
- The explicit instantiation definition directive for a **`__global__`** function template is now supported
- Fixed an issue in the CUDA compiler related to incorrect parameter pack expansion.
- The CUDA compiler previously incorrectly determined that the constructor for a **`__shared__`** multidimensional array variable was non-empty in some scenarios, and generated a spurious diagnostic. The bug has now been fixed.

4.2.2. CUDA Profiler

- Fixed an issue in **`nvprof`** where the **`--trace api`** option does not print the API calls when the **`--metrics`** option or the **`--events`** option is also specified.
- The NVLink topology diagram available in the Visual Profiler may be garbled and the rectangles representing the CPUs and GPUs may be overlapped. You can manually select and rearrange the processor rectangles to get a better layout.
- Fixed an issue in the Visual Profiler when no events or metrics are collected when profiling on a remote system.

4.2.3. CUDA Profiling Tools Interface (CUPTI)

- Fixed an issue with incorrect reporting of the half precision functional unit utilization (**`hp_fu_utilization`**) metric in CUPTI.

4.2.4. CUDA-GDB

- Fixed an issue in CUDA-GDB to where **`info float`** would trigger an assert inside a CUDA stack frame.
- Fixed an issue with CUDA-GDB where in some cases, continuing from a deleted breakpoint would result in an error on GPUs based on the Volta architecture.
- Fixed an issue with CUDA-GDB where it would crash with an OpenMP 4.5 offload program compiled with the Clang compiler.

4.2.5. CUDA-MEMCHECK

- Fixed an issue with CUDA-MEMCHECK where it did not correctly detect illegal memory accesses on GPUs based on the Volta architecture.

4.3. CUDA Libraries

4.3.1. cuBLAS Library

- Fixed an issue with a cuBLAS malfunction for specific **`int8`** row-major GEMM sizes, which resulted in incorrect results.
- Fixed an incorrect data type for **`const float`** used in batched GEMM APIs from **`const float* foo[]`** to **`const float* const foo[]`**. This fix enables users to bind a pointer of type **`float**`** or **`float*[]`** to the argument.
- Fixed the cuBLAS code sample "Application Using C and CUBLAS: 0-based Indexing" that was cut off in the PDF version of *cuBLAS Library User Guide*.

4.3.2. NVIDIA Performance Primitives (NPP)

- Fixed a functional correctness issue for the following NPP routines
  - **`nppiDilate_8u_C1R`**
  - **`nppiDilate_16u_C1R`**

Chapter 5.
KNOWN ISSUES

5.1. General CUDA

- On Ubuntu 17.10 systems, the CUDA Toolkit runfile installer may fail to install due to missing parts of the GLVND stack. Refer to the list of available GLVND packages by clicking https://packages.ubuntu.com/source/artful/libglvnd. You may need to install the following packages:
  - **`libglvnd0`**
  - **`libgl1`**
  - **`libglx0`**
  - **`libegl1`**
  - **`libgles2`**
  - **`libopengl0`**
- The driver that is supplied with CUDA 9.2 (R396) has known issues with the upcoming Windows 10 RS4 release. Users of Windows 10 RS4 should upgrade to the latest GA driver from nvidia.com.
- The **`nanosleep`** PTX instruction for Volta is not supported in this release of CUDA. It may be fully supported in a future release of CUDA. There may be references to **`nanosleep`** in the compiler headers (e.g. **`include/crt/sm_70_rt*`**). Developers are encouraged to not use this instruction in their CUDA applications on Volta until it is fully supported.
- In some cases on Windows, when CUDA 9.2 is installed with custom installation settings (where all display driver features are disabled), the existing desktop context menu may not show the **NVIDIA Display Control Panel** any more. Re-install the NVIDIA driver to obtain the control panel.
- On systems with Fedora 27, the CUDA Toolkit runfile installer may fail to install without the **`elfutils-libelf-devel`** package installed. Install the missing package or install the **`dkms`** package to complete the installation of the CUDA Toolkit.
- For warp matrix functions in this release, all threads in a warp must call the same **`load_matrix_sync()`** function at the same source line, otherwise the code

execution is likely to hang or produce unintended side effects. For example, the following usage is not supported:

```
if (threadIdx.x % 2) {
…
load_matrix_sync(...);
…
}
else {
…
load_matrix_sync(...);
…
}
```

The same restriction applies to calls to **`store_matrix_sync() and mma_sync()`**.

5.2. CUDA Tools

5.2.1. CUDA Profiler

- Tracing and profiling (event and metric collection) is not supported for multidevice cooperative kernels, that is, kernels launched by using the API functions **`cudaLaunchCooperativeKernelMultiDevice`** or **`cuLaunchCooperativeKernelMultiDevice`**.
- Because of the low resolution of the timer on Windows, the start and end timestamps can be same for activities having short execution duration on Windows. As a result, the Visual Profiler and **`nvprof`** report the following warning:
  Found *N* invalid records in the result.
- The source file for unified memory profiling results cannot be opened in the source view if the user is remote profiling on a POWER platform through Visual Profiler.
- Running both the analysis and the application in **Analysis All** fails on TCC. To work around this issue, run each unguided analysis and application analysis individually.

5.2.2. CUDA-MEMCHECK

For known issues in CUDA-MEMCHECK, see Known Issues in the CUDA-MEMCHECK documentation.

5.3. CUDA Libraries

5.3.1. cuBLAS Library

- The previously documented behavior of cuBLAS allowed the same handle to be used simultaneously from multiple host threads. However, there are multiple known issues with this, including in application crashes in some instances, and performance degradations in other situations. To avoid this issue, each host thread should use a separate cuBLAS handle to call the APIs. The documentation for the cuBLAS library has also been changed to indicate that simultaneous use of the

same handle from multiple host threads is disallowed, as the functionality and performance issues will not be addressed.
- A silent error might occur in **`cublasGemmEx`** when called with **`CUBLAS_GEMM_ALGO2_TENSOR_OP`**. We recommend applications to not use **`CUBLAS_GEMM_ALGO2_TENSOR_OP`** until the issue is fixed.

5.4. CUDA Samples

- The NVRTC samples on Mac OS do not link correctly. To work around the issue, modify the linker command in the Makefile to pass **`-L/Developer/NVIDIA/CUDA-9.2/lib`**.

Chapter 6. TEGRA SOC BASED PRODUCT RELEASE NOTES

This section of the release notes contains only information that 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 Tegra SoC based products. On Tegra SoC based products, the CUDA Toolkit version is 9.2.78.

6.1. New Features

CUDA Tegra Driver

- Support has been added for Pegasus on Vibrante Linux.
- EGL Stream has been enhanced as follows:
  - Support for additional color formats for EGL streams has been added.
  - In addition to allowing the release of frames in order, support for out-of-order release of frames has been added. Applications can use this feature to speed up their computational tasks.
- GPU work submission latency on Android, L4T, and QNX platforms has been optimized.
- Support has been added on Linux for registering host allocation, which enables the use of third-party memory to be processed directly by the GPU.

CUDA Tegra Driver API

- **`cudaDevAttrHostRegisterSupported`** now checks whether the device supports host memory registration through the **`cudaHostRegister`** API. The attribute will be set to 1 if the device supports host memory registration (beyond Xavier with kernel driver and OS support) and 0 otherwise.

6.2. Known Issues and Limitations

CUDA Tegra Driver

- Starting from CUDA 9.2, 32-bit support will no longer be available.
- During initialization, the driver reserves a large amount of CPU virtual address (VA) for its internal memory management. On QNX, this CPU VA 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 compared with earlier releases. NVIDIA is working with its partners to address this issue in upcoming releases.
- **`cudaHostRegister`** on QNX is not supported because of lack of support from the QNX kernels. This functionality will be enabled in future releases.
- CUDA allocators cannot make a single allocation greater than 4 GB on Tegra SoC memory. This limitation applies to all allocations on Tegra iGPU and zero-copy memory allocations on Tegra dGPU. To work around this limitation, ensure that applications make multiple allocations and aggregate them to create a large allocation.
- The **`cudaDeviceGetAttribute`** method returns incorrect information (false) for the attribute **`cudaDevAttrHostNativeAtomicSupported`**. Native atomics have been supported fromT194 onwards, but the device attribute is returned incorrectly.

CUDA Profiler

- 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.

CUDA-GDB

- **QNX:** `cuda-qnx-gdb` may intermittently miss device breakpoints.
- **QNX :** The `info threads` command in `cuda-qnx-gdb` displays the host threads even when the focus is on the device.
- **Linux:** CUDA-GDB may intermittently miss device exceptions.
- **Linux:** The `set cuda memcheck on` command in CUDA-GDB does not have any effect.
- **Linux:** CUDA-GDB may intermittently miss device breakpoints in CUDA applications that use the iGPU and the dGPU at the same 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.

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

Copyright

www.nvidia.com