NVIDIA CUDA TOOLKIT 9.0.461

RN-06722-001 _v9.0 | June 2018

Release Notes for Windows, Linux, and Mac OS

TABLE OF CONTENTS

ERRATA

New Features

General CUDA

- CUDA 9 now supports Multi-Process Service (MPS) on Volta GPUs. For information about enhancements to Volta MPS, see Multi-Process Service (http://docs.nvidia.com/deploy/mps/) in the NVIDIA GPU Deployment and Management Documentation.
- A code sample for the new CUDA C++ warp matrix functions has been added.
- Improved CUDA kernel launch latency (by up to 12x) when using the triple angle bracket syntax for both single and multi-threaded cases.

CUDA Tools

- **CUDA Compilers.** Microsoft Visual Studio 2017 (starting with Update 1) support is beta. Only the RTM version (vc15.0) is fully supported.
- **CUDA Compilers.** An attempt to define a **`__global__`** function in a friend declaration now generates an NVCC diagnostic. Previously, compilation would fail at the host compilation step.

CUDA Libraries

- cuBLAS 9.0.333 is an update to CUDA Toolkit 9 that improves GEMM computation performance on Tesla V100 systems and includes bug fixes aimed at deep learning and scientific computing applications. The update includes optimized performance of the **`cublasGemmEx()`** API for GEMM input sizes used in deep learning applications, such as convolutional sequence to sequence (seq2seq) models, when the **`CUBLAS_GEMM_DEFAULT_TENSOR_OP`** and **`CUBLAS_GEMM_DEFAULT`** algorithm types are used.
- cuBLAS 9.0.282 is an update to CUDA Toolkit 9 that includes GEMM performance enhancements on Tesla V100 and several bug fixes targeted for both deep learning and scientific computing applications. Key highlights of the update include:
  - Overall performance enhancements across key input sizes that are used in recurrent neural networks (RNNs) and speech models

- Optimized performance for small-tiled GEMMs with support for new HMMA and FFMA GEMM kernels
- Improved heuristics to speed up GEMMs across various input sizes

Unsupported Features

General CUDA

- **CUDA library.** The built-in functions **`__float2half_rn()`** and **`__half2float()`** have been removed. Use equivalent functionality in the updated fp16 header file from the CUDA toolkit.
- **CUDA library.** cuBLAS GemmEx routines, namely **`cublas<t>gemm`** extensions for mixed precision, are supported only on GPUs based on the Maxwell or later architectures. These routines are not supported on GPUs based on the Kepler architecture, namely Tesla K40 or K80.
- The environment variable for disabling unified memory, **`CUDA_DISABLE_UNIFIED_MEMORY`**, is no longer supported.

Resolved Issues

General CUDA

- MPS Server returns an exit status of 1 when it successfully exits.
- The performance of **`cudaLaunchCooperativeKernelMultiDevice()`** APIs has been improved.
- Strict alias warnings when using GCC to compile code that uses **`__half`** data types (`cuda_fp16.h`) have been disabled.
- The Warp Intrinsics **`__shfl()`** function for FP16 data types now has a **`*_sync`** equivalent.

CUDA Libraries

- The **`cufftXtMalloc()`** API now allocates the correct amount of memory for multi-GPU 2D and 3D plans.

Known Issues

General CUDA

- System configurations that combine IBM POWER9 CPUs with NVIDIA Volta GPUs have the hardware capability for two GPUs to map each other's memory even if there's no direct NVLink connection between those two GPUs. This feature will not be exposed in CUDA 9.0, but is planned for a future release. When supported, this feature will cause two GPUs that are not directly connected through NVLink to be reported by as being peer capable by **`cudaDeviceCanAccessPeer`**. This change could potentially affect application

behavior for applications that were developed with a driver released for CUDA 9.0 when the driver is upgraded to a driver that supports this feature.

- An update from an RPM or Debian package driver installation that includes the diagnostic driver packages to a driver installation that does not include the diagnostic driver packages may fail. To avoid this issue, remove the diagnostic driver package **before** updating the driver package. How to remove the package depends on the format of the package. For example, to remove the RPM package, use the following command:

```
yum remove xorg-x11-drv-nvidia-diagnostic
```

CUDA Tools

- PC sampling with the CUDA profiler (**`nvprof`**) can result in errors or hangs when used in GPU instances on Microsoft Azure.
- Microsoft Visual Studio 2017 (starting with Update 1) support is beta. Only the RTM version (vc15.0) is supported. Users may experience some issues when Visual Studio 2017 (starting with Update 1) is used as a host compiler with nvcc.
- When a large number of samples are collected, the Visual Profiler might not be able to show NVLink events on the timeline. To work around this issue, do one of the following:
  - Refresh the timeline by zooming in or zooming out.
  - Save and open the session.
- If you are trying to create Microsoft Visual Studio projects on Windows systems with multiple versions of CUDA installed (for example, CUDA 8 and CUDA 9), switching between CUDA runtimes by selecting **File** > **New** > **Project** > **Templates** > **NVIDIA** may result in a failure to create the project. To work around this issue, reinstall the CUDA toolkit version needed to create the project.

CUDA Libraries

- Some **`testNPP`** samples fail to compile with CUDA 9. These samples will be fixed in a future release of CUDA.
- cuBLAS Device API routines may frequently fail or not work with CUDA 9 on Tesla V100 systems. cuBLAS Device API failures are planned to be addressed in the next CUDA release.

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.

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-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

2.1. General CUDA

- CUDA 9.0 adds support for the Volta architecture.
- CUDA 9.0 adds support for new extensions to the CUDA programming model, namely, Cooperative Groups.
- CUDA 9.0 adds an API to create a CUDA event from an **`EGLSyncKHR`** object.
- CUDA 9 now supports Multi-Process Service (MPS) on Volta GPUs. For information about enhancements to Volta MPS, see Multi-Process Service (http://docs.nvidia.com/deploy/mps/) in the NVIDIA GPU Deployment and Management Documentation.
- For Linux and Windows TCC, scrub on free is enabled by default.
- The definitions of the **`__half`** and **`__half2`** data types have been updated to be semantically similar to **`float`** and **`float2`**, and **`double`** and **`double2`**. This change allows programs to use **`half`**, **`float`**, and **`double`** in natural code as algorithm precision requires, for example, by allowing templated arguments to be any floating point type. This change is purely semantic and has no impact on performance. The new behavior is implemented by using C++ operator overloads defined in the `cuda_fp16.h` header file.

  Note that the new types are incompatible with the previous CUDA 8.0 definitions. As a result, this change is a breaking change for code that uses these types in certain ways. Such code may fail at compilation time or run time. Only the type definitions are changing: all existing FP16 intrinsics, for example, **`__hadd(x, y)`**, remain supported and code that uses intrinsics will be unaffected.

  Refer to the CUDA documentation for more information about the **`__half`** and **`__half2`** data types.
- A code sample for the new CUDA C++ warp matrix functions has been added.
- Improved CUDA kernel launch latency (by up to 12x) when using the triple angle bracket syntax for both single and multi-threaded cases.

2.2. CUDA Tools

2.2.1. CUDA Compilers

- CUDA now supports the Clang 3.9 LLVM-based C and C++ compilers as host compilers on Linux operating systems.
- The CUDA compiler now supports CUDA C++ and PTX matrix-multiply operations that may be accelerated by specialized GPU hardware, specifically Tensor Cores, in the Volta architecture. This support is available as an experimental feature. For more information, refer to *CUDA C Programming Guide*.
- If an incorrect **__global__** template instantiation is launched, the compiler does not exit with return code 1, but induces CUDART to set appropriate error code.
- The CUDA compiler now supports the **-std=c++03** C++ dialect mode. This mode was added to enable existing code to be compiled with **gcc** 6.0 or later or a recent Clang compiler in **c++03** mode if necessary. The default mode for these compilers is **c++14**. An application might have to be built in **c++03** mode because it is linking to a third-party library compiled in that mode.
- The CUDA compiler can now be directed to generate warnings with the **-Wreorder** option or errors with the **-Werror=reorder** option when member initializers are reordered.
- The Intel C++ compiler 17.0 is now supported.
- The **libdevice** binaries have been consolidated into a single binary that works for all the supported GPU architectures.
- NVVM now supports functions that take a variable number or type of arguments (varargs functions).
- The CUDA compiler now supports C++14 features.
- The CUDA compiler now supports the **deprecated** attribute and declspec for references from device code. For details, refer to *CUDA Programming Guide*.
- The implementation texture and surface functions has been refactored to reduce the amount of code in implicitly included header files. This change significantly reduces the compilation time for small programs.
- The compiler has transitioned to a new code-generation back end for Kepler GPUs. PTXAS now includes a new option **--new-sm3x-opt=false** that allows developers to continue using the legacy back end. Use **ptxas --help** to get more information about these command-line options.
- GCC 6.x is now supported.
- PGI pcg++ 17 is now supported.
- Xcode 8.3.3 is now supported.
- Microsoft Visual Studio 2017 (only the RTM version vc15.0) is supported. Support of updates to Visual Studio 2017 (starting with Update 1) is beta.
- An attempt to define a **__global__** function in a friend declaration now generates an NVCC diagnostic. Previously, compilation would fail at the host compilation step.

2.3. CUDA Libraries

- New library meta packages on Linux allow users to install only the CUDA libraries without other toolkit components. Installing only the libraries reduces the overall installer footprint for library users. The following new meta packages are introduced:

  `cuda-libraries`
  This meta package contains only the library binary files, that is, all the **`.so`** files.

  `cuda-libraries-dev`
  This meta package contains library binary files and header files.

2.3.1. cuBLAS Library

- cuBLAS 9.0.333 is an update to CUDA Toolkit 9 that improves GEMM computation performance on Tesla V100 systems and includes bug fixes aimed at deep learning and scientific computing applications. The update includes optimized performance of the **`cublasGemmEx()`** API for GEMM input sizes used in deep learning applications, such as convolutional sequence to sequence (seq2seq) models, when the **`CUBLAS_GEMM_DEFAULT_TENSOR_OP`** and **`CUBLAS_GEMM_DEFAULT`** algorithm types are used.
- cuBLAS 9.0.282 is an update to CUDA Toolkit 9 that includes GEMM performance enhancements on Tesla V100 and several bug fixes targeted for both deep learning and scientific computing applications. Key highlights of the update include:
  - Overall performance enhancements across key input sizes that are used in recurrent neural networks (RNNs) and speech models
  - Optimized performance for small-tiled GEMMs with support for new HMMA and FFMA GEMM kernels
  - Improved heuristics to speed up GEMMs across various input sizes
- The Volta architecture is supported, including optimized single-precision and mixed-precision Generalized Matrix-Matrix Multiply (GEMM) matrices, namely: SGEMM and SGEMMEx with FP16 input and FP32 computation for Tesla V100 Tensor Cores.
- Performance enhancements have been made to GEMM matrices that are primarily used in deep learning applications based on Recurrent Neural Networks (RNNs) and Fully Connected Networks (FCNs).
- GEMM heuristics are improved to choose the most optimized GEMM kernel for the input matrices. Heuristics for batched GEMM matrices are also fixed.
- OpenAI GEMM kernels and optimizations in GEMM for small matrices and batch sizes have been integrated. These improvements are transparent with no API changes.

Limitations on New Features of the cuBLAS Library in CUDA 9

- Batching GEMM matrices for Tesla V100 Tensor Cores is not supported. You may not be able to use **cublas<t>gemmBatched()** APIs on Tesla V100 GPUs with Tensor Cores, but these functions will use the legacy FMA and HFMA instructions.
- Some GEMM heuristic optimizations and OpenAI GEMM kernels for small matrices are not available on Tesla V100 Tensor Cores.
- For **cublasSetMathMode()**, when set to **CUBLAS_TENSOR_OP_MATH**, **cublasSgemm()**, **cublasGemmEx()**, and **cublasSgemmEx()** will allow the Tensor Cores to be used when A/B types are set to **CUDA_R_32F**.
  - In the CUDA 9 RC build, the behavior was to perform a down conversion from FP32 to FP16 with Round to Zero.
  - In the production release of CUDA 9, the behavior is to perform a down conversion with Round to Nearest instead.
- To use single-precision and mixed-precision GEMM operations on Tensor Cores, you must do the following:

  1. Set the math mode to **Tensor_OP** by using the following function call:
     ```
     cublasSetMathMode(handle, CUBLAS_TENSOR_OP_MATH);
     ```
  2. Set the algorithm to **CUBLAS_GEMM_DFALT_TENSOR_OP** in the **cublasGemmEx** API.

  The following example shows how to use the GEMMEx API for Tensor Cores.

```
cublasSetMathMode(handle, CUBLAS_TENSOR_OP_MATH);

status = cublasGemmEx(handle,
                      CUBLAS_OP_N, CUBLAS_OP_N, N, N, N, &alpha, d_A,
                      CUDA_R_32F ,N, d_B,CUDA_R_32F, N, &beta, d_C,
                      CUDA_R_32F, N, CUDA_R_32F,
                      CUBLAS_GEMM_DFALT_TENSOR_OP);
```

2.3.2. NVIDIA Performance Primitives (NPP)

- In CUDA 9, the NPP library provides major performance improvements on image-processing and signal-processing functions.
- New primitives have been added to increase the total number of functions in the library to over 6,000.
- A single monolithic library has been split into smaller functional groups to reduce the memory footprint of the library.
- Image batching is supported for some functions to increase the overall throughput.

2.3.3. cuFFT Library

- Performance has been improved for various input sizes on single-GPU and multi-GPU environments.

2.3.4. cuSOLVER Library

- Dense eigenvalue and Singular Value Decomposition (SVD) based on the Jacobi Method are now supported. The scientific community can now use cuSOLVER to accurately and quickly solve eigenvalues with a user-specified tolerance.

2.3.5. cuSPARSE Library

- Dense and sparse matrix pruning are included to allow deep learning networks to safely omit a small percentage of values to improve performance without impacting overall accuracy.

2.3.6. nvGRAPH Library

- New algorithms to help solve key performance challenges for graph analytics applications are introduced:
  - Breadth-first search (BFS) can be used to detect or validate connected components in a graph or to find shortest paths between nodes.
  - Maximum modularity clustering can be used to analyze graphs.
- The library now supports triangle counting, graph extraction, and contraction routines for various applications such as community detection, cyber security, and ad technology.

Chapter 3. UNSUPPORTED FEATURES

The following features are officially unsupported in the current release. Developers must employ alternative solutions to these features in their software.

General CUDA

- **Fermi Architecture Support.** The CUDA Toolkit, including the CUDA compiler (**`nvcc`**), developer tools, and CUDA libraries, no longer supports the Fermi architecture (**`sm_2.x`**). Note that support for the Fermi architecture is being removed from the CUDA Toolkit but not from the driver. Applications compiled with CUDA 8 or earlier will continue to work on the Fermi architecture with newer NVIDIA drivers.
- **CUDA library.** The built-in functions **`__float2half_rn()`** and **`__half2float()`** have been removed. Use equivalent functionality in the updated fp16 header file from the CUDA toolkit.
- The environment variable for disabling unified memory, **`CUDA_DISABLE_UNIFIED_MEMORY`**, is no longer supported.

CUDA Tools

- **CUDA Compilers.** The CUDA compiler no longer supports the PTXAS option **`-abi=no`**.
- **CUDA Compilers.** The **`__CUDACC_VER__`** macro is no longer supported. The following macros are still supported and can be used instead:
  - **`__CUDACC_VER_MAJOR__`**
  - **`__CUDACC_VER_MINOR__`**
  - **`__CUDACC_VER_BUILD__`**
- **Debugger.** The **`cuda-gdb`** debugger is no longer included in the CUDA Toolkit installer packages for Mac OS.

Chapter 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

- **Warp Intrinsics.** The following warp functions are deprecated:
  - `__shfl()` (all variants)
  - `__any()`
  - `any()`
  - `__all()`
  - `all()`
  - `__ballot()`

  These functions have been replaced with new synchronizing equivalent functions with the `*_sync` suffix. The new warp intrinsics take a mask of threads that explicitly define which lanes (threads of a warp) must participate in the warp intrinsic. For more information, refer to *CUDA C Programming Guide*.
- **Windows Server 2008 R2 Support.** Support for Windows Server 2008 R2 is now deprecated and will be removed in a future version of the CUDA Toolkit.

CUDA Tools

- **32-bit Linux CUDA Applications.** CUDA Toolkit support for 32-bit Linux CUDA applications has been dropped. Existing 32-bit applications will continue to work with the 64-bit driver, but support is deprecated.

Chapter 5. RESOLVED ISSUES

5.1. General CUDA

- The following device functions have been removed:
  - **`syncthreads()`**. Use **`__syncthreads()`** instead.
  - **`trap()`**. Use **`__trap()`** instead.
  - **`brkpt()`**. Use **`__brkpt()`** instead.
  - **`prof_trigger()`**. Use **`__prof_trigger()`** instead.
  - **`threadfence()`**. Use **`__threadfence()`** instead.

  The functions to use instead provide identical functionality.
- When using the runfile installer on a system for which DRM is not enabled on the system kernel, extract the driver runfile with the **`--extract`** option and install the driver separately with the **`--no-drm`** option.
- The Mac OS installer is missing uninstallation scripts.
- On Windows systems that do not have Microsoft Visual Studio 2017 installed, the Nsight Visual Studio Edition Summary page may indicate that Nsight for Visual Studio 2017 was installed.
- MPS Server returns an exit status of 1 when it successfully exits.
- The performance of **`cudaLaunchCooperativeKernelMultiDevice()`** APIs has been improved.
- Strict alias warnings when using GCC to compile code that uses **`__half`** data types (`cuda_fp16.h`) have been disabled.

5.2. CUDA Tools

5.2.1. CUDA Compilers

- Kernel launch doesn't find global functions through ADL.

5.2.2. CUDA Profiler

- In the Visual Profiler, the NVLink Analysis diagram may be incorrect after the diagram is scrolled. This error can be corrected by horizontally resizing the diagram panel.
- In the Visual Profiler, the **Run** > **Configure Metrics and Events** dialog box does not work for the device that has NVLink support. To work around this issue, collect all metrics and events by using **`nvprof`** and then import them into **`nvvp`**.
- Profiling applications using **`nvprof`** or Visual Profiler on systems without an NVIDIA driver resulted in an error. This is now reported as a warning.

5.2.3. CUDA Profiling Tools Interface (CUPTI)

- Some devices with compute capability 6.1 don't support multicontext scope collection for metrics.
- The timestamp and duration for some memory copies on some devices with compute capability 6.1 are incorrect. This issue can result in errors, and dependency analysis results may not be available. This issue can impact **`nvprof`**, **`nvvp`**, and **`cupti`**.

5.3. CUDA Libraries

5.3.1. cuFFT Library

- The **`cufftXtMalloc()`** API now allocates the correct amount of memory for multi-GPU 2D and 3D plans.

Chapter 6. KNOWN ISSUES

6.1. General CUDA

- On Mac OS, CUDA binaries that target the Volta architecture (**`compute_70`** or **`sm_70`**) cannot be built with the CUDA 9 compiler (**`nvcc`**). This issue will be fixed in a future release.
- If the Windows toolkit installation fails, it may be because Visual Studio, **`Nvda.Launcher.exe`**, **`Nsight.Monitor.exe`**, or **`Nvda.CrashReporter.exe`** is running. Make sure these programs are closed and try to install again.
- Many OpenCL tests of OpenMM functionality fail on all GPUs.
- When **`CUDA_VISIBLE_DEVICES`** is 0, many applications fail with **`cudaErrorMemoryAllocation`**.
- On Windows, when a user deselects driver components in the custom installation page, other components in that UI form may disappear. To work around this issue, click the area where the components should appear to force them to reappear.
- System configurations that combine IBM POWER9 CPUs with NVIDIA Volta GPUs have the hardware capability for two GPUs to map each other's memory even if there's no direct NVLink connection between those two GPUs. This feature will not be exposed in CUDA 9.0, but is planned for a future release. When supported, this feature will cause two GPUs that are not directly connected through NVLink to be reported by as being peer capable by **`cudaDeviceCanAccessPeer`**. This change could potentially affect application behavior for applications that were developed with a driver released for CUDA 9.0 when the driver is upgraded to a driver that supports this feature.
- An update from an RPM or Debian package driver installation that includes the diagnostic driver packages to a driver installation that does not include the diagnostic driver packages may fail. To avoid this issue, remove the diagnostic driver package **before** updating the driver package. How to remove the package depends on the format of the package. For example, to remove the RPM package, use the following command:

```
yum remove xorg-x11-drv-nvidia-diagnostic
```

6.2. CUDA Tools

6.2.1. CUDA Compiler

- Microsoft Visual Studio 2017 (starting with Update 1) support is beta. Only the RTM version (vc15.0) is supported. Users may experience some issues when Visual Studio 2017 (starting with Update 1) is used as a host compiler with nvcc.

6.2.2. CUDA-GDB

- On Mac OS, the desktop image becomes corrupted when a CUDA application stops at device-code break point.
- On Mac OS, single-GPU debugging on GM204/Titan and Quadro K6000 with ECC disabled caused the system to hang.

6.2.3. CUDA Profiler

- On Windows, using the mouse wheel to scroll does not work within the Visual Profiler.
- PC sampling with the CUDA profiler (**`nvprof`**) can result in errors or hangs when used in GPU instances on Microsoft Azure.
- When a large number of samples are collected, the Visual Profiler might not be able to show NVLink events on the timeline. To work around this issue, do one of the following:
  - Refresh the timeline by zooming in or zooming out.
  - Save and open the session.
- If you are trying to create Microsoft Visual Studio projects on Windows systems with multiple versions of CUDA installed (for example, CUDA 8 and CUDA 9), switching between CUDA runtimes by selecting **File** > **New** > **Project** > **Templates** > **NVIDIA** may result in a failure to create the project. To work around this issue, reinstall the CUDA toolkit version needed to create the project.

6.2.4. Nsight Eclipse Edition

- Nsight MSI needed for CUDA 9.0 EA.

6.2.5. CUDA Samples

- On Ubuntu 17.04, the **`3_Imaging/cudaDecodeGL`** sample fails to build. To work around this issue, use the following command when building **`cudaDecodeGL`**:

```
make EXTRA_CCFLAGS=-no-pie
```

- OpenCL samples dump core.
- Several samples fail if **`CUDA_VISIBLE_DEVICES`** sets a Quadro GPU as the display output.

6.3. CUDA Libraries

6.3.1. NVIDIA Performance Primitives (NPP)

- Some **`testNPP`** samples fail to compile with CUDA 9. These samples will be fixed in a future release of CUDA.

6.3.2. cuBLAS Library

- cuBLAS Device API routines may frequently fail or not work with CUDA 9 on Tesla V100 systems. cuBLAS Device API failures are planned to be addressed in the next CUDA release.

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

Trademarks

Copyright

www.nvidia.com