NVIDIA CUDA TOOLKIT 9.1.246

RN-06722-001 _v9.1 | April 2018

Release Notes for Windows, Linux, and Mac OS

TABLE OF CONTENTS

ERRATA

New Features

General CUDA

- Added support for the IBM Power LE (POWER9) platform
- The CUDA Memory Operations API, which is used by applications that take advantage of GPUDirect, is disabled by default and can be enabled only on Linux. As a result of this change:
  - The **`cuDeviceGetAttribute(CU_DEVICE_ATTRIBUTE_CAN_USE_STREAM_MEM_OPS)`** function returns false.
  - All memory-operation functions return **`CUDA_ERROR_NOT_SUPPORTED`**.

  Note that the memory operations APIs can be enabled only on Linux. To do so, do the following:

  ```
  modprobe nvidia NVreg_EnableStreamMemOPs=1
  ```
- Driver support for Mac OS 10.13.3 is added in this release.
- Xcode 9.2 is now supported as a host compiler on Mac OS.

CUDA Libraries

- cuBLAS 9.1.181 is an update to CUDA Toolkit 9.1 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 9.1.128 is an update to CUDA Toolkit 9.1 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

Resolved Issues

General CUDA

- Issues in **`p2pBandwidthLatencyTest`** have been resolved. P2P accesses are enabled in both directions and the latency test has been changed to perform write operations instead of read operations.
- A memory allocation issue in the CUDA driver on GPUs based on architectures earlier than Pascal, such as Kepler and Maxwell, has been resolved. This issue occurred with certain CUDA workloads, for example, heavy-use cases of RNN tests. In these cases, subsequent CUDA kernels sometimes failed to launch with error code 30.

CUDA Tools

- **CUDA Compilers.** An issue in the compiler in separate compilation mode where, in some cases, divergent threads in a warp may not synchronize correctly at a **`__syncwarp`** instruction has been resolved.
- **CUDA Compilers.** The update to the PTX assembler (**`ptxas`** 9.1.121) fixes an issue affecting the compilation of code that performs address calculations using large immediate operands.
- **CUDA Compilers.** The update to the PTX assembler (**`ptxas`** 9.1.121) fixes an issue where, in some cases, the unrolling of loops and subsequent optimizations may cause incorrect array element updates.

CUDA Libraries

- Fixed an issue with **`cusparse<t>csrmv_mp()`** that could result in errors on some matrix sizes.

Known Issues

General CUDA

- The **`CU_STREAM_WAIT_VALUE_FLUSH`** and **`CU_STREAM_MEM_OP_FLUSH_REMOTE_WRITES`** memory operation flags are disabled in this release. When the hardware supports these flags, a device attribute to enable them will be made available in a future release of CUDA.

CUDA Tools

- **CUDA Compiler.** In some cases, when NVVM IR is compiled with **`libNVVM`** on GCC with debugging information (**`-g`**), **`ptxas`** may fail with the message: `Parsing error near '-': syntax error.`
- **CUDA Compiler.** Explicit instantiation definition directive for a **`__global__`** function template is not supported.

- **CUDA Compiler.** In some extended-precision code sequences, the addition operators may not produce the carry bit. This issue may cause different results to be generated on Volta architectures compared to other GPU architectures. To work around this bug, use a newer driver (R390 or higher) and perform just in time compilation (JIT) on the code sequence.
- **CUDA Compiler.** 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()`**.
- **CUDA-GDB.** The version information reported by CUDA **`gdbserver`** is “9.0” when it should be “9.1”.
- **CUDA Profiler.** When multiple sessions are being used with CUDA Visual Profiler, the session connection settings for one session might be modified to the latest connection settings used in another session. To work around this issue after changing a session, you may need to choose the appropriate connection under the **Settings** view.
- **CUDA Profiler.** On POWER9 systems, the Visual Profiler NVLink topology diagram may be garbled and the rectangles representing the CPUs and GPUs may be overlapped. To get a better layout, you can manually select and rearrange the processor rectangles.

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 and earlier, at https://github.com/NVIDIA/cuda-gdb.
- Upon request by sending an e-mail to mailto:oss-requests@nvidia.com.

Chapter 2. NEW FEATURES

2.1. General CUDA

- Added support for the IBM Power LE (POWER9) platform
- Full core-dump generation for all clients when using MPS on GPUs based on the Volta architecture is now supported.
- Platform support for SUSE SLES12 SP3 has been added in this release.
- Platform support for OpenSUSE Leap 42.3 has been added in this release.
- The CUDA Memory Operations API, which is used by applications that take advantage of GPUDirect, is disabled by default and can be enabled only on Linux. As a result of this change:

  - The **`cuDeviceGetAttribute(CU_DEVICE_ATTRIBUTE_CAN_USE_STREAM_MEM_OPS)`** function returns false.
  - All memory-operation functions return **`CUDA_ERROR_NOT_SUPPORTED`**.

  Note that the memory operations APIs can be enabled only on Linux. To do so, do the following:

  ```
  modprobe nvidia NVreg_EnableStreamMemOPs=1
  ```
- CUDA now supports the following additional matrix shapes in C++ warp matrix operations (WMMA):

  - 8m×32n×16k
  - 32m×8n×16k

  For more information, refer to Warp matrix functions (http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#wmma) in *CUDA C Programming Guide*.
- Driver support for Mac OS 10.13.3 is added in this release.
- Xcode 9.2 is now supported as a host compiler on Mac OS.

2.2. CUDA Tools

2.2.1. CUDA Compilers

- XLC 13.1.6 is now supported as a host compiler on ppc64le on Linux.
- A new restriction has been introduced: Operator functions cannot be marked `__global__`. The CUDA front end will now generate an error for this scenario. Previously, compilation would complete in some instances, and fail in host compiler invocation with others, for example, template operator functions marked `__global__`.
- Clang 4 is now supported as a host compiler on x86_64 and POWER (ppc64le) on Linux.
- `nvdisasm` now optionally includes instruction offsets while printing the CFG. The command line option `--print-instr-offsets-cfg` (`-poff`) to turn on this feature has been added.
- C++ 14 features are now supported when the Intel C++ compiler 17.0 is used as a host compiler for `nvcc`.

2.2.2. CUDA-GDB

- CUDA now supports lightweight core dumps as a preview feature. To enable this feature, set the following environment variables:
  - `CUDA_ENABLE_LIGHTWEIGHT_COREDUMP=1`
  - `CUDA_ENABLE_COREDUMP_ON_EXCEPTION=1`

2.3. CUDA Libraries

- CUB 1.7.4 has been integrated as a device back end for Thrust.

2.3.1. cuBLAS Library

- cuBLAS 9.1.181 is an update to CUDA Toolkit 9.1 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 9.1.128 is an update to CUDA Toolkit 9.1 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
- Two functions have been added to improve deep learning performance on GPUs based on the Volta architecture. These functions perform matrix-matrix multiplication of a series of matrices with mixed-precision input, output, and compute formats. They are an extension to the existing batched GEMM API, which now includes the ability to specify mixed-precision formats. You can now take advantage of Tensor Cores on Tesla V100 GPUs to perform batched GEMM computation on 16-bit floating point input and output formats, and use 32-bit floating format for computation. Note that these new functions are available only on GPUs with compute capability >=5.0. For details of these new functions, refer to *cuBLAS Library User Guide* (http://docs.nvidia.com/cuda/cublas/).

2.3.2. NVIDIA Performance Primitives (NPP)

- New image augmentation routines are added for deep learning, including routines for the following purposes:
    - Connected image marker labeling
    - Marker label compression
    - Bound segments

  The new image augmentation routines also include a set of morphology functions:
    - Gray Dilate Border
    - Gray Erode Border
    - Morph Close Border
    - Morph Open Border
    - Morph Top Hat Border
    - Morph Black Hat Border
    - Morph Gradient Border

  For more information, refer to the documentation for NPP (http://docs.nvidia.com/cuda/npp/).

2.3.3. cuFFT Library

- Large 2D and 3D FFT sizes are now enabled on multi-GPU systems. For example, 3D 1K FFTs can be run on a system with four NVIDIA® Tesla® V100 GPUs. The library includes optimized heuristics to lower the overall memory usage.
- **`cuFFTW`** now supports device memory allocated by **`cudaMalloc`** and **`cudaMallocManaged`**.
- **`cuFFTW`** APIs now include support for CUDA unified memory to improve performance of OpenACC workloads.

Chapter 3. PERFORMANCE IMPROVEMENTS

3.1. General CUDA

- The CUDA kernel launch latency has been reduced by a factor of up to 12 when the triple angle bracket syntax for both single-threaded and multithreaded cases is used.

3.2. CUDA Tools

3.2.1. cuSOLVER Library

- Eigensolver performance optimizations using tridiagonalization (**`sytrd`**) enable higher performance for chemistry workloads, and simulation of large compounds and molecules. For matrices with up to 256 elements, performance is up to for times faster. For matrices with 2048 or more elements, performance is up to 40% faster.

3.3. CUDA Libraries

3.3.1. cuFFT Library

- New optimizations reduce memory usage for 2D/3D input sizes. cuFFT will automatically switch to lower-memory version of 2D and 3D algorithms when the available memory is limited. This behavior enables larger FFT sizes on multi-GPU systems for power of two sizes.
- The cuFFT library now optimizes performance and memory footprint by always using 32-bit indexing for some FFTs that span more than 4 billion elements.

Chapter 4. RESOLVED ISSUES

4.1. General CUDA

- The toolkit cluster packages on Linux no longer include the local installer repositories for the NVIDIA driver.
- Users no longer need to reboot after installing the CUDA Driver on Mac 10.12 to run CUDA applications.
- 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 should no longer fail. However, to perform such an update, you must follow the instructions in Advanced Setup (http://docs.nvidia.com/cuda/cuda-installation-guide-linux/index.html#advanced-setup) in *NVIDIA CUDA Installation Guide for Linux*.
- An issue related to the reading of user passwords by the distribution-independent CUDA installer (runfile) has been resolved.
- When installing just the driver on Fedora using RPMs, you no longer need to install using the **`nvidia-drivers`** meta-package rather than the **`cuda-drivers`** meta-package.
- An issue that caused OpenCL tests of OpenMM functionality to fail on all GPUs has been resolved.
- An issue that caused many applications to fail with **`cudaErrorMemoryAllocation`** when **`CUDA_VISIBLE_DEVICES`** is 0 has been resolved.
- Issues in **`p2pBandwidthLatencyTest`** have been resolved. P2P accesses are enabled in both directions and the latency test has been changed to perform write operations instead of read operations.
- A memory allocation issue in the CUDA driver on GPUs based on architectures earlier than Pascal, such as Kepler and Maxwell, has been resolved. This issue occurred with certain CUDA workloads, for example, heavy-use cases of RNN tests. In these cases, subsequent CUDA kernels sometimes failed to launch with error code 30.

4.2. CUDA Tools

4.2.1. CUDA Compilers

- An issue with **`ptxas`** when **`ptxas`** checks function declaration for errors, which caused **`ptxas`** to crash when a mismatch was detected, has been resolved.
- An issue with declaring multidimensional arrays using constant expressions has been resolved.
- An issue where **`cuda-memcheck`** may hang on GPUs based on the Volta architecture with applications that make heavy use of templated code has been resolved.
- An issue in the compiler in separate compilation mode where, in some cases, divergent threads in a warp may not synchronize correctly at a **`__syncwarp`** instruction has been resolved.
- The update to the PTX assembler (**`ptxas`** 9.1.121) fixes an issue affecting the compilation of code that performs address calculations using large immediate operands.
- The update to the PTX assembler (**`ptxas`** 9.1.121) fixes an issue where, in some cases, the unrolling of loops and subsequent optimizations may cause incorrect array element updates.

4.2.2. CUDA Profiler

- 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.
- 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.
- The CUDA profiler (**`nvprof`**) no longer generates spurious warnings when used with the **`--analysis-metrics`** option.
- PC sampling with the CUDA profiler (**`nvprof`**) can result in errors or hangs when used in GPU instances on Microsoft Azure.
- The CUDA profiler (**`nvprof`**) can now be used with several MPI ranks using a shared temporary directory specified with the **`$TMPDIR`** environment variable.
- An issue with PC sampling using the Visual Profiler for multiblock cooperative group APIs has now been resolved.

4.2.3. CUDA-MEMCHECK

- The CUDA-MEMCHECK tool now correctly detects illegal memory accesses on GPUs based on the Volta architecture.

4.3. CUDA Libraries

4.3.1. NVIDIA Performance Primitives (NPP)

- Some **`testNPP`** samples that failed to compiled with CUDA 9 now compile.

4.3.2. cuSPARSE Library

- Fixed an issue with **`cusparse<t>csrmv_mp()`** that could result in errors on some matrix sizes.

4.4. CUDA Samples

- The **`3_Imaging/EGLStream_CUDA_CrossGPU`** and **`3_Imaging/EGLStreams_CUDA_Interop`** samples fail to build with old Khronos EGL headers. The samples can be made to compile by installing the latest Khronos EGL headers. Alternatively, when building all the samples with the global-level **`Makefile`**, pass the **`-k`** option to continue building the remainder of the samples.
- An issue that caused several samples to fail if **`CUDA_VISIBLE_DEVICES`** sets a Quadro GPU as the display output has been resolved.

Chapter 5. KNOWN ISSUES

5.1. General CUDA

- 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.
- 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 9.1, 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 9.1 when the driver is upgraded to a driver that supports this feature.
- The **`CU_STREAM_WAIT_VALUE_FLUSH`** and **`CU_STREAM_MEM_OP_FLUSH_REMOTE_WRITES`** memory operation flags are disabled in this release. When the hardware supports these flags, a device attribute to enable them will be made available in a future release of CUDA.

5.2. CUDA Tools

5.2.1. CUDA Compiler

- In some cases, when NVVM IR is compiled with **`libNVVM`** on GCC with debugging information (**`-g`**), **`ptxas`** may fail with the message: `Parsing error near '-': syntax error`.
- Explicit instantiation definition directive for a **`__global__`** function template is not supported.
- In some extended-precision code sequences, the addition operators may not produce the carry bit. This issue may cause different results to be generated on Volta

architectures compared to other GPU architectures. To work around this bug, use a newer driver (R390 or higher) and perform just in time compilation (JIT) on the code sequence.

- 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()`**.

5.2.2. CUDA-GDB

- The version information reported by CUDA **`gdbserver`** is “9.0” when it should be “9.1”.

5.2.3. CUDA Profiler

- When multiple sessions are being used with CUDA Visual Profiler, the session connection settings for one session might be modified to the latest connection settings used in another session. To work around this issue after changing a session, you may need to choose the appropriate connection under the **Settings** view.
- On POWER9 systems, the Visual Profiler NVLink topology diagram may be garbled and the rectangles representing the CPUs and GPUs may be overlapped. To get a better layout, you can manually select and rearrange the processor rectangles.

5.2.4. CUDA Profiling Tools Interface (CUPTI)

- Access to PM registers during profiling can result in errors.

5.2.5. 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
```

5.3. CUDA Libraries

5.3.1. cuBLAS Library

- The following functions are not supported on the device cuBLAS API library (**`cublas_device.a`**):

  - **`cublas<t>gemmBatched()`**
  - **`cublasBatchedGemmEx`**
  - **`cublasGemmExStridedBatched`**

  Any attempt to use these functions with the device API library will result in a link error. For more details about cuBLAS library functions or limitations, refer to *cuBLAS Library User Guide* (http://docs.nvidia.com/cuda/cublas/).

5.3.2. cuFFT Library

- There is a known issue with certain cuFFT plans that causes an assertion in the execution phase of certain plans. This applies to plans with all of the following characteristics: real input to complex output (R2C), in-place, native compatibility mode, certain even transform sizes, and more than one batch.

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.