NVIDIA CUDA Toolkit

Release Notes for CUDA 11.2

Table of Contents

List of Tables

Chapter 1. CUDA 11.2 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.

1.1. CUDA Toolkit Major Component Versions

CUDA Components

Starting with CUDA 11, the various components in the toolkit are versioned independently.

For CUDA 11.2, the table below indicates the versions:

Table 1. CUDA 11.2 Component Versions

Component Name	Version Information	Supported Architectures
CUDA Runtime (cudart)	11.2.146	x86_64, POWER, Arm64
cuobjdump	11.2.135	x86_64, POWER, Arm64
CUPTI	11.2.135	x86_64, POWER, Arm64
CUDA cuxxfilt (demangler)	11.2.135	x86_64, POWER, Arm64
CUDA Demo Suite	11.2.67	x86_64
CUDA GDB	11.2.135	x86_64, POWER, Arm64
CUDA Memcheck	11.2.135	x86_64, POWER
CUDA NVCC	11.2.142	x86_64, POWER, Arm64
CUDA nvdisasm	11.2.135	x86_64, POWER, Arm64
CUDA NVML Headers	11.2.67	x86_64, POWER, Arm64
CUDA nvprof	11.2.135	x86_64, POWER, Arm64
CUDA nvprune	11.2.135	x86_64, POWER, Arm64
CUDA NVRTC	11.2.142	x86_64, POWER, Arm64
CUDA NVTX	11.2.67	x86_64, POWER, Arm64
CUDA NVVP	11.2.135	x86_64, POWER
CUDA Samples	11.2.135	x86_64, POWER, Arm64

Component Name	Version Information	Supported Architectures
CUDA Compute Sanitizer API	11.2.135	x86_64, POWER, Arm64
CUDA cuBLAS	11.4.1.1026	x86_64, POWER, Arm64
CUDA cuFFT	10.4.0.135	x86_64, POWER, Arm64
CUDA cuRAND	10.2.3.135	x86_64, POWER, Arm64
CUDA cuSOLVER	11.1.0.135	x86_64, POWER, Arm64
CUDA cuSPARSE	11.4.0.135	x86_64, POWER, Arm64
CUDA NPP	11.3.2.139	x86_64, POWER, Arm64
CUDA nvJPEG	11.4.0.135	x86_64, POWER, Arm64
Nsight Compute	2020.3.1.3	x86_64, POWER, Arm64
Nsight Windows NVTX	1.21018621	x86_64, POWER, Arm64
Nsight Systems	2020.4.3.7	x86_64, POWER, Arm64
Nsight Visual Studio Edition (VSE)	2020.3.1.21012	x86_64 (Windows)
NVIDIA Linux Driver	460.32.03	x86_64, POWER, Arm64
NVIDIA Windows Driver	461.09	x86_64 (Windows)

CUDA Driver

Running a CUDA application requires the system with at least one CUDA capable GPU and a driver that is compatible with the CUDA Toolkit. See Table 2. 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.

Note: Starting with CUDA 11.0, the toolkit components are individually versioned, and the toolkit itself is versioned as shown in the table below.

Table 2. CUDA Toolkit and Minimum Compatible Driver Versions

CUDA Toolkit	Linux x86_64 Driver Version	Windows x86_64 Driver Version
CUDA 11.2.1 Update 1	>=460.32.03	>=461.09
CUDA 11.2.0 GA	>=460.27.03	>=460.82
CUDA 11.1.1 Update 1	>=455.32	>=456.81
CUDA 11.1 GA	>=455.23	>=456.38
CUDA 11.0.3 Update 1	>= 450.51.06	>= 451.82
CUDA 11.0.2 GA	>= 450.51.05	>= 451.48
CUDA 11.0.1 RC	>= 450.36.06	>= 451.22

CUDA Toolkit	Linux x86_64 Driver Version	Windows x86_64 Driver Version
CUDA 10.2.89	>= 440.33	>= 441.22
CUDA 10.1 (10.1.105 general release, and updates)	>= 418.39	>= 418.96
CUDA 10.0.130	>= 410.48	>= 411.31
CUDA 9.2 (9.2.148 Update 1)	>= 396.37	>= 398.26
CUDA 9.2 (9.2.88)	>= 396.26	>= 397.44
CUDA 9.1 (9.1.85)	>= 390.46	>= 391.29
CUDA 9.0 (9.0.76)	>= 384.81	>= 385.54
CUDA 8.0 (8.0.61 GA2)	>= 375.26	>= 376.51
CUDA 8.0 (8.0.44)	>= 367.48	>= 369.30
CUDA 7.5 (7.5.16)	>= 352.31	>= 353.66
CUDA 7.0 (7.0.28)	>= 346.46	>= 347.62

For convenience, the NVIDIA driver is installed as part of the CUDA Toolkit installation. Note that this driver is for development purposes and is not recommended for use in production with Tesla GPUs.

For running CUDA applications in production with Tesla GPUs, it is recommended to download the latest driver for Tesla GPUs from the NVIDIA driver downloads site at http://www.nvidia.com/drivers.

During the installation of the CUDA Toolkit, the installation of the NVIDIA driver may be skipped on Windows (when using the interactive or silent installation) or on Linux (by using meta packages).

For more information on customizing the install process on Windows, see http://docs.nvidia.com/cuda/cuda-installation-guide-microsoft-windows/index.html#install-cuda-software.

For meta packages on Linux, see https://docs.nvidia.com/cuda/cuda-installation-guide-linux/index.html#package-manager-metas

1.2. What's New in CUDA 11.2.1

This section summarizes the changes in CUDA 11.2.1 (11.2 Update 1) since the 11.2.0 GA release.

CUDA Compiler

- Resolved Issues:
  - Previously, when using recent versions of VS 2019 host compiler, a call to `pow(double, int)` or `pow(float, int)` in host or device code sometimes caused build failures.This issue has been resolved.

cuSOLVER

- New Features:
  - New singular value decomposition (GESVDR) is added. GESVDR computes partial spectrum with random sampling, an order of magnitude faster than GESVD.
  - `libcusolver.so` no longer links `libcublas_static.a`; instead, it depends on `libcublas.so`. This reduces the binary size of `libcusolver.so`. However, it breaks backward compatibility. The user has to link `libcusolver.so` with the correct version of `libcublas.so`.

cuSPARSE

- New Features:
  - New Tensor Core-accelerated Block Sparse Matrix - Matrix Multiplication (`cusparseSpMM`) and introduction of the Blocked-Ellpack storage format.
  - New algorithms for CSR/COO Sparse Matrix - Vector Multiplication (`cusparseSpMV`) with better performance.
  - Extended functionalities for `cusparseSpMV`:
    - Support for the CSC format.
    - Support for regular/complex bfloat16 data types for both uniform and mixed-precision computation.
    - Support for mixed regular-complex data type computation.
    - Support for deterministic and non-deterministic computation.
  - New algorithm (`CUSPARSE_SPMM_CSR_ALG3`) for Sparse Matrix - Matrix Multiplication (`cusparseSpMM`) with better performance especially for small matrices.
  - New routine for Sampled Dense Matrix - Dense Matrix Multiplication (`cusparseSDDMM`) which deprecated `cusparseConstrainedGeMM` and provides better performance.
  - Better accuracy of `cusparseAxpby`, `cusparseRot`, `cusparseSpVV` for bfloat16 and half regular/complex data types.
  - All routines support NVTX annotation for enhancing the profiler time line on complex applications.
- Deprecations:
  - `cusparseConstrainedGeMM` has been deprecated in favor of `cusparseSDDMM`.
  - `cusparseCsrmvEx` has been deprecated in favor of `cusparseSpMV`.
  - COO Array of Structure (CooAoS) format has been deprecated including `cusparseCreateCooAoS`, `cusparseCooAoSGet`, and its support for `cusparseSpMV`.
- Known Issues:
  - `cusparseDestroySpVec`, `cusparseDestroyDnVec`, `cusparseDestroySpMat`, `cusparseDestroyDnMat`, `cusparseDestroy` with `NULL` argument could cause segmentation fault on Windows.
- Resolved Issues:
  - `cusparseAxpby`, `cusparseGather`, `cusparseScatter`, `cusparseRot`, `cusparseSpVV`, `cusparseSpMV` now support zero-size matrices.
  - `cusparseCsr2cscEx2` now correctly handles empty matrices (nnz = 0).

  - `cusparseXcsr2csr_compress` now uses 2-norm for the comparison of complex values instead of only the real part.

NPP

- New features:
  - New APIs added to compute Distance Transform using Parallel Banding Algorithm (PBA):
    - `nppiDistanceTransformPBA_`*`xxxxx`*`_C1R_Ctx()` – where *xxxxx* specifies the input and output combination: 8u16u, 8s16u, 16u16u, 16s16u, 8u32f, 8s32f, 16u32f, 16s32f
    - `nppiSignedDistanceTransformPBA_32f_C1R_Ctx()`
- Resolved issues:
  - Fixed the issue in which Label Markers adds zero pixel as object region.

NVJPEG

- New Features:
  - nvJPEG decoder added a new API to support region of interest (ROI) based decoding for batched hardware decoder:
    - `nvjpegDecodeBatchedEx()`
    - `nvjpegDecodeBatchedSupportedEx()`

cuFFT

- Known Issues:
  - cuFFT planning and plan estimation functions may not restore correct context affecting CUDA driver API applications.
  - Plans with strides, primes larger than 127 in FFT size decomposition and total size of transform including strides bigger than 32GB produce incorrect results.
- Resolved Issues:
  - Previously, reduced performance of power-of-2 single precision FFTs was observed on GPUs with sm_86 architecture. This issue has been resolved.
  - Large prime factors in size decomposition and real to complex or complex to real FFT type no longer cause cuFFT plan functions to fail.

CUPTI

- Deprecations early notice:

  The following functions are scheduled to be deprecated in 11.3 and will be removed in a future release:

  - `NVPW_MetricsContext_RunScript` and `NVPW_MetricsContext_ExecScript_Begin` from the header `nvperf_host.h`.
  - `cuptiDeviceGetTimestamp` from the header `cupti_events.h`

1.3. General CUDA

- Added support for RHEL 7.9, RHEL 8.3, Fedora 33 and Debian 10.6 Buster on x86_64 platforms.
- Stream ordered memory allocator: Added new APIs `cudaMallocAsync()` and `cudaFreeAsync()` to enable applications to order memory allocation and deallocation with other work launched into a CUDA stream. Provides significant performance improvements compared to `cudaMalloc()`/`cudaFree()`. Introduces the concept of memory pools to provide the application with more control over memory management. Each device has a default memory pool and custom memory pools can be created as needed.
- Added support for importing DirectX11/12 textures with format `DXGI_FORMAT_NV12` via the CUDA external resource interoperability APIs. For more details about external resource interoperability API functions, see https://docs.nvidia.com/cuda/cuda-runtime-api/group__CUDART__EXTRES__INTEROP.html#group__CUDART__EXTRES__INTEROP.
- Added new Driver and Runtime API functions, `cuArrayGetPlane` and `cudaArrayGetPlane` respectively, to get individual format plane CUDA arrays from multi-planar formatted CUDA arrays.
- Two new API functions have been added to get the list of architectures supported by the NVRTC library:
  - `nvrtcGetNumSupportedArchs`
  - `nvrtcGetSupportedArchs`
- CUDA Graph enhancements:
  - Two new graph node types: external semaphore signal and external semaphore wait. This allows new types of synchronization between graph workloads and non-CUDA workloads. Enables support for explicitly adding these nodes to the graph as well as adding nodes by capturing calls to `cuSignalExternalSemaphoresAsync()`, `cudaSignalExternalSemaphoresAsync()`, `cuWaitExternalSemaphoresAsync()`, and `cudaWaitExternalSemaphoresAsync()`.

    New APIs:

    - `cudaGraphAddExternalSemaphoresSignalNode()`
    - `cudaGraphExternalSemaphoresSignalNodeGetParams()`
    - `cudaGraphExternalSemaphoresSignalNodeSetParams()`
    - `cudaGraphAddExternalSemaphoresWaitNode()`
    - `cudaGraphExternalSemaphoresWaitNodeGetParams()`
    - `cudaGraphExternalSemaphoresWaitNodeSetParams()`
    - `cudaGraphExecExternalSemaphoresSignalNodeSetParams()`
    - `cudaGraphExecExternalSemaphoresWaitNodeSetParams()`

  - Support for updating the function of an instantiated kernel node via explicit node update using cudaGraphExecKernelNodeSetParams or whole graph update cudaGraphExecUpdate APIs.
  - Improved device-side performance for the first launch of a graph following instantiation or updates.
- Cooperative groups enhancement: `cudaLaunchCooperativeKernel` now enables simultaneous launch in multiple streams allowing multiple co-operative grids to run concurrently.
- Extended the external semaphore interface to work with Vulkan Timeline Semaphores on Linux and Windows platforms. Enables support to import, wait, and signal on Vulkan Timeline Semaphores exported from Vulkan via `cuImportExternalSemaphore()`, `cudaImportExternalSemaphore()`, `cuSignalExternalSemaphoresAsync()`, `cudaSignalExternalSemaphoresAsync()`, `cuWaitExternalSemaphoresAsync()`, and `cudaWaitExternalSemaphoresAsync()`. On Linux, semaphores are imported from an FD exported from a Vulkan semaphore. On Windows, semaphores are imported via an NT handle exported from a Vulkan semaphore.
- CUDA Runtime: Protoype for `cudaGetFuncBySymbol` changed to add `cdecl` qualifier.

1.4. CUDA Tools

1.4.1. CUDA Compilers

- nvcc has added an option `--optimization-info=inline`. Using this option, diagnostics are emitted about inlining decisions made by the compiler for device code. For functions not inlined, additional information about the reason for not inlining are emitted. The compiler may invoke inlining pass multiple times and a callsite not inlined in an earlier pass may be inlined in a future pass.
- Added nvcc options:
  - `--display-error-number` to display error number for warnings
  - `--diag-error errNum,...` to override the severity of errEnum,... to errors
  - `--diag-suppress errNum,...` to suppress warnings errEnum,...
  - `--diag-warn errNum,...` to override the severity of errEnum,... to warning
- nvcc now provides support for the following builtin functions, for providing optimization hints to the compiler:
  - `__builtin_assume()`
  - `__assume()`
  - `__builtin_assume_aligned()`

  In addition, pre-existing support for `__builtin_expect()` is now documented. See Compiler Optimization Hint Functions for details.

1.4.2. CUDA Tools

- A preview version of a new tool, cu++filt, is included in this release. NVCC produces mangled names, appearing in PTX files, which do not strictly follow the mangling conventions of the Itanium ABI--and are thus not properly demangled by standard tools such as binutils' c++filt. Specifically, this is true for PTX function parameters. The new cu ++filt utility will demangle all of these correctly. As this is a preview version of the utility, feedback is invited. For more information, see cu++filt.

1.4.3. CUDA Developer Tools

- For changes to nvprof and Visual Profiler, see the changelog.
- For new features, improvements, and bug fixes in CUPTI, see the changelog.
- For new features, improvements, and bug fixes in Nsight Compute, see the changelog.

1.5. CUDA Libraries

1.5.1. cuFFT Library

- Multi-GPU plans can be associated with a stream using the `cufftSetStream` API function call.
- Performance improvements for R2C/C2C/C2R transforms.
- Performance improvements for multi-GPU systems.

1.6. Deprecated or Removed Features

The following features are deprecated or removed in the current release of the CUDA software. Deprecated features still work in the current release, but their documentation may have been removed, and they will become officially unsupported in a future release. We recommend that developers employ alternative solutions to these features in their software.

General CUDA

- Support for Visual Studio versions <= 2015 has been dropped in CUDA 11.2, following its deprecation in CUDA 11.1.

1.7. Resolved Issues

1.7.1. cuFFT Library

- cuFFT is no longer stuck in a bad state if previous plan creation fails with `CUFFT_ALLOC_FAILED`.

- Previously, single dimensional multi-GPU FFT plans ignored user input on `cufftXtSetGPUs` *whichGPUs* argument and assumed that GPUs IDs are always numbered from 0 to N-1. This issue has been resolved.
- Plans with primes larger than 127 in FFT size decomposition or FFT size being a prime number bigger than 4093 do not perform calculations on second and subsequent `cufftExecute*` calls. Regression was introduced in cuFFT 11.1

1.7.2. cuSOLVER Library

- `cusolverDnIRSXgels` sometimes returned `CUSOLVER_STATUS_INTERNAL_ERROR` when the precision is 'z'. This issue has been fixed in CUDA 11.2; now `cusolverDnIRSXgels` works for all precisions.
- `ZSYTRF` sometimes returned `CUSOLVER_STATUS_INTERNAL_ERROR` due to insufficient resources to launch the kernel. This issue has been fixed in CUDA 11.2
- GETRF returned early without finishing the whole factorization when the matrix was singular. This issue has been fixed in CUDA 11.2

1.8. Known Issues

1.8.1. General CUDA

- Upgrading the driver on SUSE or SLES may result in a failure to load the NVIDIA kernel modules. This occurs specifically when the G04 packages are upgraded to the G05 packages. To remedy this, run the following to force a reinstall of the KMP (kernel module package):

```
$ sudo zypper in --force `rpm -qa "nvidia-gfx*kmp*"
```

1.8.2. cuBLAS Library

- `cublas<s/d/c/z>Gemm()` with very large n and m=k=1 may fail on Pascal devices.

1.8.3. cuFFT Library

- cuFFT planning and plan estimation functions may not restore correct context affecting CUDA driver API applications.

1.8.4. cuSPARSE Library

- `cusparseXdense2csr` provides incorrect results for some matrix sizes.

Notice

VESA DisplayPort

HDMI

OpenCL

Trademarks

Copyright

NVIDIA Corporation | 2788 San Tomas Expressway, Santa Clara, CA 95051
www.nvidia.com