NVIDIA CUDA Toolkit 11.2.76

Release Notes for CUDA 11.2
# Table of Contents

Chapter 1. CUDA 11.2 Release Notes..................................................................................1
  1.1. CUDA Toolkit Major Component Versions.................................................................1
  1.2. General CUDA...........................................................................................................3
  1.3. CUDA Tools................................................................................................................5
    1.3.1. CUDA Compilers....................................................................................................5
    1.3.2. CUDA Tools............................................................................................................5
    1.3.3. CUDA Developer Tools........................................................................................5
  1.4. CUDA Libraries..........................................................................................................6
    1.4.1. cuFFT Library........................................................................................................6
  1.5. Deprecated or Removed Features..............................................................................6
  1.6. Resolved Issues..........................................................................................................6
    1.6.1. cuFFT Library........................................................................................................6
    1.6.2. cuSOLVER Library................................................................................................6
  1.7. Known Issues.............................................................................................................7
    1.7.1. General CUDA......................................................................................................7
    1.7.2. CUDA Compiler....................................................................................................7
    1.7.3. cuBLAS Library....................................................................................................7
    1.7.4. cuFFT Library........................................................................................................7
    1.7.5. cuSPARSE Library................................................................................................7
List of Tables

Table 1. CUDA 11.2 Component Versions ......................................................................................... 1
Table 2. CUDA Toolkit and Compatible Driver Versions ................................................................. 2
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

<table>
<thead>
<tr>
<th>Component Name</th>
<th>Version Information</th>
<th>Supported Architectures</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUDA Runtime (cudart)</td>
<td>11.2.72</td>
<td>x86_64, POWER, Arm64</td>
</tr>
<tr>
<td>cuobjdump</td>
<td>11.2.67</td>
<td>x86_64, POWER, Arm64</td>
</tr>
<tr>
<td>CUPTI</td>
<td>11.2.67</td>
<td>x86_64, POWER, Arm64</td>
</tr>
<tr>
<td>CUDA cuxxfilt {demangler}</td>
<td>11.2.67</td>
<td>x86_64, POWER, Arm64</td>
</tr>
<tr>
<td>CUDA Demo Suite</td>
<td>11.2.67</td>
<td>x86_64</td>
</tr>
<tr>
<td>CUDA GDB</td>
<td>11.2.67</td>
<td>x86_64, POWER, Arm64</td>
</tr>
<tr>
<td>CUDA Memcheck</td>
<td>11.2.67</td>
<td>x86_64, POWER</td>
</tr>
<tr>
<td>CUDA NVCC</td>
<td>11.2.67</td>
<td>x86_64, POWER, Arm64</td>
</tr>
<tr>
<td>CUDA nvdisasm</td>
<td>11.2.67</td>
<td>x86_64, POWER, Arm64</td>
</tr>
<tr>
<td>CUDA NVML Headers</td>
<td>11.2.67</td>
<td>x86_64, POWER, Arm64</td>
</tr>
<tr>
<td>CUDA nvprof</td>
<td>11.2.67</td>
<td>x86_64, POWER, Arm64</td>
</tr>
<tr>
<td>CUDA nvprune</td>
<td>11.2.67</td>
<td>x86_64, POWER, Arm64</td>
</tr>
<tr>
<td>CUDA NVRTC</td>
<td>11.2.67</td>
<td>x86_64, POWER, Arm64</td>
</tr>
<tr>
<td>CUDA NVTX</td>
<td>11.2.67</td>
<td>x86_64, POWER, Arm64</td>
</tr>
<tr>
<td>CUDA NWVP</td>
<td>11.2.67</td>
<td>x86_64, POWER</td>
</tr>
<tr>
<td>CUDA Samples</td>
<td>11.2.67</td>
<td>x86_64, POWER, Arm64</td>
</tr>
</tbody>
</table>
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.


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 Compatible Driver Versions

<table>
<thead>
<tr>
<th>CUDA Toolkit</th>
<th>Linux x86_64 Driver Version</th>
<th>Windows x86_64 Driver Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUDA 11.2.0 GA</td>
<td>&gt;=460.27.03</td>
<td>&gt;=460.82</td>
</tr>
<tr>
<td>CUDA 11.1.1 Update 1</td>
<td>&gt;=455.32</td>
<td>&gt;=456.81</td>
</tr>
<tr>
<td>CUDA 11.1 GA</td>
<td>&gt;=455.23</td>
<td>&gt;=456.38</td>
</tr>
<tr>
<td>CUDA 11.0.3 Update 1</td>
<td>&gt;=450.51.06</td>
<td>&gt;=451.82</td>
</tr>
<tr>
<td>CUDA 11.0.2 GA</td>
<td>&gt;=450.51.05</td>
<td>&gt;=451.48</td>
</tr>
<tr>
<td>CUDA 11.0.1 RC</td>
<td>&gt;=450.36.06</td>
<td>&gt;=451.22</td>
</tr>
</tbody>
</table>
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. 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](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: Prototype for `cudaGetFuncBySymbol` changed to add `cdecl` qualifier.

1.3. CUDA Tools

1.3.1. 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 built-in 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.

1.3.2. 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.

1.3.3. 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.
1.4. CUDA Libraries

1.4.1. 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.

1.5. 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.

1.6. Resolved Issues

1.6.1. 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

1.6.2. 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
CUDA 11.2 Release Notes

1.7. Known Issues

1.7.1. General CUDA

- GETRF returned early without finishing the whole factorization when the matrix was singular. This issue has been fixed in CUDA 11.2.

1.7.2. CUDA Compiler

- Windows, when using recent versions of VS 2019 host compiler, a call to `pow(double, int)` or `pow(float, int)` in host or device code may cause build failure. This is an NVCC issue. For example:

```
//--
int main() {
    double x = pow(1.0, 1);
    return (int)x;
}
//--
```

The above program may fail to build with a linker error in CUDA 11.2, when using recent versions of VS 2019 host compiler. A source workaround is to ensure that both arguments are of `double` type, e.g. `pow(1.0, (double)1)`.

1.7.3. cuBLAS Library

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

1.7.4. cuFFT Library

- Plans with strides, primes larger than 127 in FFT size decomposition and total size of transform including strides bigger than 32GB produce incorrect results.
- Reduced performance of power-of-2 single precision FFTs on GPUs with sm_86 architecture.
- cuFFT planning and plan estimation functions may not restore correct context affecting CUDA driver API applications.

1.7.5. cuSPARSE Library

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

This document is provided for information purposes only and shall not be regarded as a warranty of a certain functionality, condition, or quality of a product. NVIDIA Corporation ("NVIDIA") makes no representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assumes no responsibility for any errors contained herein. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This document is not a commitment to develop, release, or deliver any Material (defined below), code, or functionality.

NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice.

Customer should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer ("Terms of Sale"). NVIDIA hereby expressly objects to applying any customer general terms and conditions with regards to the purchase of the NVIDIA product referenced in this document. No contractual obligations are formed either directly or indirectly by this document.

VESA DisplayPort

DisplayPort and DisplayPort Compliance Logo, DisplayPort Compliance Logo for Dual-mode Sources, and DisplayPort Compliance Logo for Active Cables are trademarks owned by the Video Electronics Standards Association in the United States and other countries.

HDMI

HDMI, the HDMI logo, and High-Definition Multimedia Interface are trademarks or registered trademarks of HDMI Licensing LLC.

OpenCL

OpenCL is a trademark of Apple Inc. used under license to the Khronos Group Inc.

Trademarks

NVIDIA and the NVIDIA logo are trademarks or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright

© 2007-2020 NVIDIA Corporation. All rights reserved.