NVSHMEM

Release Notes

RN-08515-001_v2.5.0 | April 2022

Table of Contents

Chapter 1. NVSHMEM Release 2.5.0

This is the NVIDIA® NVSHMEM™ 2.5.0 release notes.

Key Features And Enhancements

This NVSHMEM release includes the following key features and enhancements:

- Added multi-instance support in NVSHMEM.

  NVSHMEM now builds as two libraries, `libnvshmem_host.so` and `libnvshmem_device.a`, which allows an application to have multiple components (for example, shared libraries and the application) that use NVSHMEM.

  

  Note: Support for the `libnvshmem.a` library still exists for legacy purposes but will be eventually removed.

- Added the `nvshmemx_init_status` API to query the initialized state of NVSHMEM.
- Added support for `CUDA_VISIBLE_DEVICES`.

  Support for `CUDA_VISIBLE_DEVICES` is not yet available with CUDA VMM, so you must set `NVSHMEM_DISABLE_CUDA_VMM=1`.
- Updated PMI and PMI-2 bootstraps to plug-ins.
- Added the `nvshmem-info` utility to display information about the NVSHMEM library.
- Fixed warnings when using NVSHMEM in applications that compile without the Relocatable Device Code (RDC) option.
- Renamed internal variables to avoid potential conflicts with variables in the application.
- Implemented the `nvshmem_alltoallmem` API.
- Improved the GPU-to-NIC assignment logic for the Summit/Sierra supercomputer.
- Fixed the host barrier API implementation for non-blocking on stream `( (*_nbi_on_stream))` point-to-point operations.
- Updated descriptions for NVSHMEM environment variables that are displayed by using `nvshmem-info` or by setting `NVSHMEM_INFO=1`.

Compatibility

NVSHMEM 2.5.0 has been tested with the following:

- CUDA:
  - 10.2
  - 11.0
  - 11.6
- On x86 and Power 9 processors

Limitations

- VMM support is disabled by default on Power 9 systems because of a performance regression.
- MPG support is not yet available on Power 9 systems.
- Systems with PCIe peer-to-peer communication require one of the following:
  - InfiniBand to support NVSHMEM atomics APIs.
  - The use of NVSHMEM's UCX transport that, if IB is absent, will use sockets for atomics.

Fixed Issues

There are no fixed issues in this release.

Breaking Changes

There are no breaking changes in this release.

Known Issues

- NVSHMEM device APIs can **only** be statically linked.

  This is because the linking of CUDA device symbols does not work across shared libraries.
- `nvshmem_barrier*`, `nvshmem_quiet`, and `nvshmem_wait_until` only ensure PE-PE ordering and visibility on systems with NVLink and InfiniBand.

  They do not ensure global ordering and visibility.
- Complex types, which are enabled by setting `NVSHMEM_COMPLEX_SUPPORT` at compile time, are not currently supported.
- When built with GDRcopy and when using Infiniband, NVSHMEM cannot allocate the complete device memory because of the inability to reuse the BAR1 space.

  This will be fixed with future CUDA driver releases in the 470 (or later) and in the 460 branch.

- When NVSHMEM maps the symmetric heap using cudaMalloc, it sets the `CU_POINTER_ATTRIBUTE_SYNC_MEMOPS` attribute, which automatically synchronizes synchronous CUDA memory operations on the symmetric heap.

  With CUDA 11.3 and later, NVSHMEM supports the mapping of the symmetric heap by using the CUDA VMM APIs. However, when you map the symmetric heap by using the VMM APIs, CUDA does not support this attribute, and users are responsible for synchronization. For additional information about synchronous CUDA memory operations, see API synchronization behavior.

Chapter 2. NVSHMEM Release 2.4.1

This is the NVIDIA® NVSHMEM™ 2.4.1 release notes.

Key Features And Enhancements

This NVSHMEM release includes the following key features and enhancements:

- Added limited support for Multiple Processes per GPU (MPG) on x86 platforms.
  - The amount of support depends on the availability of CUDA MPS.
  - MPG support is currently not available on Power 9 platforms.
- Added a local buffer registration API that allows non-symmetric buffers to be used as local buffers in the NVSHMEM API.
- Added support for dynamic symmetric heap allocation, which eliminates the need to specify `NVSHMEM_SYMMETRIC_SIZE`.
  - On x86 platforms, this feature is is **enabled** by default, and is available with CUDA version 11.3 or later.
  - On P9 platforms, this feature is **disabled** by default, and can be enabled by using the `NVSHMEM_DISABLE_CUDA_VMM` environment variable.
- Support for large RMA messages.
- To build NVSHMEM without `ibrc` support, set `NVSHMEM_IBRC_SUPPORT=0` in the environment before you build.

  This allows you to build and run NVSHMEM without the GDRCopy and OFED dependencies.
- Support for calling `nvshmem_init/finalize` multiple times with an MPI bootstrap.
- Improved testing coverage (large messages, exercising full GPU memory, and so on).
- Improved the default PE to NIC assignment for NVIDIA DGX-2™ systems.
- Optimized channel request processing by using the CPU proxy thread.
- Added support for the `shmem_global_exit` API.
- Removed redundant barriers to improve the collectives' performance.
- Significant code refactoring to use templates instead of macros for internal functions.
- Improved performance for device-side blocking RMA and strided RMA APIs.
- Bug fix for buffers with large offsets into the NVSHMEM symmetric heap.

Compatibility

NVSHMEM 2.4.1 has been tested with the following:

- CUDA:
  - 10.2
  - 11.0
  - 11.5
- On x86 and Power 9 processors

Limitations

- VMM support is disabled by default on Power 9 systems because of a performance regression.
- MPG support is not yet available on Power 9 systems.
- Systems with PCIe peer-to-peer communication require one of the following:
  - InfiniBand to support NVSHMEM atomics APIs.
  - The use of NVSHMEM's UCX transport that, if IB is absent, will use sockets for atomics.

Fixed Issues

There are no fixed issues in this release.

Breaking Changes

There are no breaking changes in this release.

Known Issues

- NVSHMEM can only be linked statically.

  This is because the linking of CUDA device symbols does not work across shared libraries.
- `nvshmem_barrier*`, `nvshmem_quiet`, and `nvshmem_wait_until` only ensure PE-PE ordering and visibility on systems with NVLink and InfiniBand.

  They do not ensure global ordering and visibility.
- Complex types, which are enabled by setting `NVSHMEM_COMPLEX_SUPPORT` at compile time, are not currently supported.
- When built with GDRcopy and when using Infiniband, NVSHMEM cannot allocate the complete device memory because of the inability to reuse the BAR1 space.

  This will be fixed with future CUDA driver releases in the 470 (or later) and in the 460 branch.

- When NVSHMEM maps the symmetric heap using cudaMalloc, it sets the `CU_POINTER_ATTRIBUTE_SYNC_MEMOPS` attribute, which automatically synchronizes synchronous CUDA memory operations on the symmetric heap.

  With CUDA 11.3 and later, NVSHMEM supports the mapping of the symmetric heap by using the CUDA VMM APIs. However, when you map the symmetric heap by using the VMM APIs, CUDA does not support this attribute, and users are responsible for synchronization. For additional information about synchronous CUDA memory operations, see API synchronization behavior.

Chapter 3. NVSHMEM Release 2.2.1

This is the NVIDIA® NVSHMEM™ 2.2.1 release notes.

Key Features And Enhancements

This NVSHMEM release includes the following key features and enhancements:

- Implemented dynamic heap memory allocation for runs with P2P GPUs.

  This feature, which requires CUDA version 11.3 or later, can be enabled by using `NVSHMEM_DISABLE_CUDA_VMM=0`. Support for IB runs will be added in the next release.
- Improved UCX transport performance for AMO and RMA operations.
- Improved performance for warp and block put/get operations.
- Added atomic support for PCIe-connected GPUs over the UCX transport.
- The UCX transport now supports non-symmetric buffers for use as local buffers in RMA and AMO operations.
- Added support to initialize NVSHMEM in CUmodule.
- Enabled MPI and PMIx bootstrap modules to be compiled externally from the NVSHMEM build.

  This allows multiple builds of these plugins to support various MPI and PMIx libraries. To select the plugins, set `NVSHMEM_BOOTSTRAP="plugin"` and `NVSHMEM_BOOTSTRAP_PLUGIN="plugin_name.so"`.

  Note: The plugin sources are installed with the compiled NVSHMEM library.
- Enabled MPI bootstrap to be used with `nvshmem_init`.

  You can set `NVSHMEM_BOOTSTRAP=MPI` or use the bootstrap plugin method.
- Fixed bugs in `nvshmem_<typename>_g` and the fetch atomics implementation.
- Changed `nvshmem_<typename>_collect` to `nvshmem_<typename>_fcollect` to match the OpenSHMEM specification.
- Fixed a type of `nreduce` argument in the reduction API to `size_t` to match OpenSHMEM specification.

- Improved NVSHMEM build times with a multi-threaded option in the CUDA compiler (requires CUDA version 11.2 and later).
- Several fixes to address Coverity reports.

Compatibility

NVSHMEM 2.2.1 has been tested with the following:

- CUDA:
  - 10.2
  - 11.0
  - 11.4
- On x86 and Power 9 processors

Limitations

Systems with PCIe peer-to-peer communication require one of the following:

- InfiniBand to support NVSHMEM atomics APIs.
- The use of NVSHMEM's UCX transport that, if IB is absent, will use sockets for atomics.

Fixed Issues

There are no fixed issues in this release.

Breaking Changes

- Changed `nvshmem_<typename>_collect` to `nvshmem_<typename>_fcollect` to match the OpenSHMEM specification.
- Fixed a type of `nreduce` argument in the `reduction` API to `size_t` to match OpenSHMEM specification.
- Removed support for host-side NVSHMEM wait APIs.

Known Issues

- NVSHMEM can only be linked statically.

  This is because the linking of CUDA device symbols does not work across shared libraries.
- `nvshmem_barrier*`, `nvshmem_quiet`, and `nvshmem_wait_until` only ensure PE-PE ordering and visibility on systems with NVLink and InfiniBand.

  They do not ensure global ordering and visibility.
- Complex types, which are enabled by setting `NVSHMEM_COMPLEX_SUPPORT` at compile time, are not currently supported.
- When built with GDRcopy and when using Infiniband, NVSHMEM cannot allocate the complete device memory because of the inability to reuse the BAR1 space.

This will be fixed with future CUDA driver releases in the 470 (or later) and in the 460 branch.

- When NVSHMEM maps the symmetric heap using cudaMalloc, it sets the `CU_POINTER_ATTRIBUTE_SYNC_MEMOPS` attribute, which automatically synchronizes synchronous CUDA memory operations on the symmetric heap.

  With CUDA 11.3 and later, NVSHMEM supports the mapping of the symmetric heap by using the CUDA VMM APIs. However, when you map the symmetric heap by using the VMM APIs, CUDA does not support this attribute, and users are responsible for synchronization. For additional information about synchronous CUDA memory operations, see API synchronization behavior.

Chapter 4. NVSHMEM Release 2.1.2

This is the NVIDIA® NVSHMEM™ 2.1.2 release notes.

Key Features And Enhancements

This NVSHMEM release includes the following key features and enhancements:

- Added a new UCX internode communication transport layer.

Note: UCX is experimental for this release.

- Added support for the automatic warp-level coalescing of `nvshmem_g` operations.
- Added support for put-with-signal operations on CUDA streams.
- Added support to map the symmetric heap by using the `cuMem` APIs.
- Improved the performance of the single-threaded NVSHMEM `put`/`get` device API.
- Added the `NVSHMEM_MAX_TEAMS` environment variable to specify the maximum number of teams that can be created.
- Improved the host and on-stream Alltoall performance by using NCCL.
- Fixed a bug in the compare-and-swap operation that caused several bytes of the compare operand to be lost.
- Improved support for single-node environments without InfiniBand.
- Added CPU core affinity to debugging output.
- Added support for the CUDA 11.3 `cudaDeviceFlushGPUDirectRDMAWrites` API for consistency.
- Improved support for the NVIDIA Tools Extension (NVTX) to enable performance analysis through NVIDIA NSight.
- Removed the `NVSHMEM_IS_P2P_RUN` environment variable, because runtime automatically determines it.
- Made improvements to NVSHMEM example codes.
- Added the `NVSHMEM_REMOTE_TRANSPORT` environment variable to select the networking layer that is used for communication between nodes.
- Set the `maxrregcount` to 32 for non-inlined device functions to ensure that calling these NVSHMEM functions does not negatively affect kernel occupancy.

Compatibility

NVSHMEM 2.1.2 has been tested with the following:

- CUDA:
  - 10.2
  - 11.0
  - 11.3
- On x86 and Power 9 processors

Limitations

Systems with PCIe peer-to-peer communication require InfiniBand to support NVSHMEM atomics APIs.

Fixed Issues

There are no fixed issues in this release.

Breaking Changes

- Removed the following deprecated constants:
  - `_NVSHMEM_MAJOR_VERSION`
  - `_NVSHMEM_MINOR_VERSION`
  - `_NVSHMEM_VENDOR_STRING`
- Removed support for the deprecated `nvshmem_wait` API.

Known Issues

- NVSHMEM can only be linked statically.

  This is because the linking of CUDA device symbols does not work across shared libraries.
- `nvshmem_barrier*`, `nvshmem_quiet`, and `nvshmem_wait_until` only ensure PE-PE ordering and visibility on systems with NVLink and InfiniBand.

  They do not ensure global ordering and visibility.
- Complex types, which are enabled by setting `NVSHMEM_COMPLEX_SUPPORT` at compile time, are not currently supported.
- In some cases, `nvshmem_<typename>_g` over InfiniBand and RoCE has been reported to return stale data.

  We are continuing to investigate this issue. In the meantime, you can use `nvshmem_<typename>_atomic_fetch` as a workaround for `nvshmem_<typename>_g`, but the performance of these options is different.

- When built with GDRcopy and when using Infiniband, NVSHMEM cannot allocate the complete device memory because of the inability to reuse the BAR1 space.

  This will be fixed with future CUDA driver releases in the 470 (or later) and in the 460 branch.
- When NVSHMEM maps the symmetric heap using cudaMalloc, it sets the `CU_POINTER_ATTRIBUTE_SYNC_MEMOPS` attribute, which automatically synchronizes synchronous CUDA memory operations on the symmetric heap.

  With CUDA 11.3 and later, NVSHMEM supports the mapping of the symmetric heap by using the CUDA VMM APIs. However, when you map the symmetric heap by using the VMM APIs, CUDA does not support this attribute, and users are responsible for synchronization. For additional information about synchronous CUDA memory operations, see API synchronization behavior.

Chapter 5. NVSHMEM Release 2.0.3

This is the NVIDIA® NVSHMEM™ 2.0.3 release notes.

Key Features And Enhancements

This NVSHMEM release includes the following key features and enhancements:

- Added the teams and team-based collectives APIs from OpenSHMEM 1.5.
- Added support to use the NVIDIA® Collective Communication Library (NCCL) for optimized NVSHMEM host and on-stream collectives.
- Added support for RDMA over Converged Ethernet (RoCE) networks.
- Added support for PMI-2 to enable an NVSHMEM job launch with srun/SLURM.
- Added support for PMIx to enable an NVSHMEM job launch with PMIx-compatible launchers, such as Slurm and Open MPI.
- Uniformly reformatted the perftest benchmark output.
- Added support for the `putmem_signal` and `signal_wait_until` APIs.
- Improved support for single-node environments without InfiniBand.
- Fixed a bug that occurred when large numbers of fetch atomic operations were performed on InfiniBand.
- Improved topology awareness in NIC-to-GPU assignments for NVIDIA® DGX™ A100 systems.
- Added the `NVSHMEM_CUDA_LIMIT_STACK_SIZE` environment variable to set the GPU thread stack size on Power systems.
- Updated the threading level support that was reported for host and stream-based APIs to `NVSHMEM_THREAD_SERIALIZED`.

  Device-side APIs support `NVSHMEM_THREAD_MULTIPLE`.

Compatibility

NVSHMEM 2.0.3 has been tested with the following:

- The following version of CUDA:
  - 10.2

  - 11.0
  - 11.1
- x86 and Power 9

Limitations

There are no limitations in this release.

Fixed Issues

- Concurrent NVSHMEM collective operations with active sets are not supported.
- Concurrent NVSHMEM memory allocation operations and collective operations are not supported.

  The OpenSHMEM specification has clarified that only memory management routines that operate on `NVSHMEM_TEAM_WORLD`, and no other collectives on that team, are permitted concurrently.

Breaking Changes

- Removed support for active set-based collectives interface in OpenSHMEM.

Known Issues

- NVSHMEM and libraries that use NVSHMEM can only be built as static libraries and not as shared libraries.

  This is because the linking of CUDA device symbols does not work across shared libraries.
- `nvshmem_barrier*`, `nvshmem_quiet`, and `nvshmem_wait_until` only ensure PE-PE ordering and visibility on systems with NVLink and InfiniBand.

  They do not ensure global ordering and visibility.
- Complex types, which are enabled by setting `NVSHMEM_COMPLEX_SUPPORT` at compile time, are not currently supported.
- In some cases, `nvshmem_<typename>_g` over InfiniBand and RoCE has been reported to return stale data.

  We are continuing to investigate this issue. In the meantime, you can use `nvshmem_<typename>_atomic_fetch` as a workaround for `nvshmem_<typename>_g`, but the performance of these options is different.

Chapter 6. NVSHMEM Release 2.0.2 EA

This is the NVIDIA® NVSHMEM™ 2.0.2 EA release notes.

Key Features And Enhancements

This NVSHMEM release includes the following key features and enhancements:

- Added the teams and team-based collectives APIs from OpenSHMEM 1.5.
- Added support to use the NVIDIA® Collective Communication Library (NCCL) for optimized NVSHMEM host and on-stream collectives.

  Note: This feature is not yet supported on Power 9 systems.

- Added support for RDMA over Converged Ethernet (RoCE) networks.
- Added support for PMI-2 to enable an NVSHMEM job launch with srun/SLURM.
- Added support for PMIx to enable an NVSHMEM job launch with PMIx-compatible launchers, such as Slurm and Open MPI.
- Uniformly reformatted the perftest benchmark output.
- Added support for the `putmem_signal` and `signal_wait_until` APIs.
- Improved support for single-node environments without InfiniBand.
- Fixed a bug that occurred when large numbers of fetch atomic operations were performed on InfiniBand.
- Improved topology awareness in NIC-to-GPU assignments for DGX A100 systems.

Compatibility

NVSHMEM 2.0.2 EA has been tested with the following:

- The following version of CUDA:
  - 10.2
  - 11.0
  - 11.1
- x86 and Power 9

Limitations

- NVSHMEM with NCCL is not yet supported on Power 9 systems.

Fixed Issues

- Concurrent NVSHMEM collective operations with active sets are not supported.
- Concurrent NVSHMEM memory allocation operations and collective operations are not supported.

  The OpenSHMEM specification has clarified that only memory management routines that operate on `NVSHMEM_TEAM_WORLD`, and no other collectives on that team, are permitted concurrently.

Breaking Changes

- Removed support for active set-based collectives interface in OpenSHMEM.

Known Issues

- NVSHMEM and libraries that use NVSHMEM can only be built as static libraries and not as shared libraries.

  This is because the linking of CUDA device symbols does not work across shared libraries.
- `nvshmem_barrier*`, `nvshmem_quiet`, and `nvshmem_wait_until` only ensure PE-PE ordering and visibility on systems with NVLink and InfiniBand.

  They do not ensure global ordering and visibility.
- Complex types, which are enabled by setting `NVSHMEM_COMPLEX_SUPPORT` at compile time, are not currently supported.
- In some cases, `nvshmem_<typename>_g` over InfiniBand and RoCE has been reported to return stale data.

  We are continuing to investigate this issue. In the meantime, you can use `nvshmem_<typename>_atomic_fetch` as a workaround for `nvshmem_<typename>_g`, but the performance of these options is different.

Chapter 7. NVSHMEM Release 1.1.3

This is the NVIDIA® NVSHMEM™ 1.1.3 release notes.

Key Features And Enhancements

This NVSHMEM release includes the following key features and enhancements:

- Implemented the `nvshmem_<type>_put_signal` API from OpenSHMEM 1.5.
- Added the `nvshmemx_signal_op` API.
- Optimized the implementation of a signal set operation over P2P connected GPUs.
- Optimized the performance of the `nvshmem_fence()` function.
- Optimized the latency of the NVSHMEM atomics API.
- Fixed a bug in the `nvshmem_ptr` API.
- Fixed a bug in the implementation of the host-side strided transfer (iput, iget,and so on) API.
- Fixed a bug in the on-stream reduction for the `long long` datatype.
- Fixed a hang during the nvshmem barrier collective operation.
- Fixed `__device__ nvshmem_quiet()` to also do quiet on IB ops to self.

Compatibility

NVSHMEM 1.1.3 has been tested with the following:

- CUDA 10.1, 10.2, and 11.0
- x86 and PowerPC

Known Issues

- NVSHMEM and libraries that use NVSHMEM can only be built as static libraries, not as shared libraries.

  This is because linking of CUDA device symbols does not work across shared libraries.
- NVSHMEM collective operations with active sets are not supported.
- Concurrent NVSHMEM memory allocation operations and collective operations are not supported.

- `nvshmem_barrier*`, `nvshmem_quiet`, and `nvshmem_wait_until` only ensure PE-PE ordering and visibility on systems with NVLink and InfiniBand.

  They do not ensure global ordering and visibility.

Chapter 8. NVSHMEM Release 1.0.1

This is the NVIDIA® NVSHMEM™ 1.0.1 release notes. This is the first official release of NVSHMEM.

Key Features And Enhancements

This NVSHMEM release includes the following key features and enhancements.

- Combines the memory of multiple GPUs into a partitioned global address space that's accessed through NVSHMEM APIs.
- Includes a low-overhead, in-kernel communication API for use by GPU threads.
- Includes stream-based and CPU-initiated communication APIs.
- Supports peer-to-peer communication using NVIDIA® NVLink® and PCI Express and for GPU clusters using NVIDIA Mellanox® InfiniBand.
- Supports x86 and POWER9 processors.
- Is interoperable with MPI and other OpenSHMEM implementations.

Compatibility

NVSHMEM 1.0.1 has been tested with the following:

- CUDA 10.1, 10.2, and 11.0 RC
- x86 and PowerPC

Known Issues

- NVSHMEM and libraries that use NVSHMEM can only be built as static libraries, not as shared libraries. This is because linking of CUDA device symbols does not work across shared libraries.
- NVSHMEM collective operations with overlapping active sets are known not to work in some scenarios.
- `nvshmem_quiet` only ensures PE-PE visibility and not global visibility of data.

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.

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.

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.

NVIDIA products are not designed, authorized, or warranted to be suitable for use in medical, military, aircraft, space, or life support equipment, nor in applications where failure or malfunction of the NVIDIA product can reasonably be expected to result in personal injury, death, or property or environmental damage. NVIDIA accepts no liability for inclusion and/or use of NVIDIA products in such equipment or applications and therefore such inclusion and/or use is at customer's own risk.

NVIDIA makes no representation or warranty that products based on this document will be suitable for any specified use. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer's sole responsibility to evaluate and determine the applicability of any information contained in this document, ensure the product is suitable and fit for the application planned by customer, and perform the necessary testing for the application in order to avoid a default of the application or the product. Weaknesses in customer's product designs may affect the quality and reliability of the NVIDIA product and may result in additional or different conditions and/or requirements beyond those contained in this document. NVIDIA accepts no liability related to any default, damage, costs, or problem which may be based on or attributable to: (i) the use of the NVIDIA product in any manner that is contrary to this document or (ii) customer product designs.

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.

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

Trademarks

NVIDIA, the NVIDIA logo, and CUDA, CUDA Toolkit, GPU, Kepler, Mellanox, NVLink, NVSHMEM, and Tesla are trademarks and/or registered trademarks of NVIDIA Corporation in the United States and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright

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