



NVIDIA HPC SDK RELEASE NOTES

RN-09976-001-V24.3 | March 2024



TABLE OF CONTENTS

Chapter 1. What's New.....	1
Chapter 2. Release Component Versions.....	2
Chapter 3. Supported Platforms.....	4
3.1. Platform Requirements for the HPC SDK.....	4
3.2. Supported CUDA Toolchain Versions.....	5
Chapter 4. Known Limitations.....	6
Chapter 5. Deprecations and Changes.....	8

LIST OF TABLES

Table 1 HPC SDK Release Components	2
Table 2 HPC SDK Platform Requirements	4

Chapter 1.

WHAT'S NEW

Welcome to the 24.3 version of the NVIDIA HPC SDK, a comprehensive suite of compilers and libraries enabling developers to program the entire HPC platform, from the GPU foundation to the CPU and out through the interconnect. The 24.3 release of the HPC SDK includes new features as well as important functionality and performance improvements.

- ▶ The HPC Compilers have implemented the `-gpu=unified` option for OpenMP Target Offload to support coherent memory on systems that use the Grace Hopper superchip. The `-gpu=unified` option now works with OpenACC, C++ and Fortran Standard Parallelism, and OpenMP, including on x86_64 systems with the HMM driver. See [the documentation](#) for more further information.
- ▶ The unified attribute has been implemented for use in CUDA Fortran applications in combination with the `-gpu=unified` on systems that support coherent memory.
- ▶ The NVIDIA Performance Libraries (NVPL) that provide essential mathematical libraries optimized for Arm 64-bit architectures are now integrated into the HPC SDK. See [NVIDIA Performance Libraries](#) for more details.

Chapter 2.

RELEASE COMPONENT VERSIONS

The NVIDIA HPC SDK 24.3 release contains the following versions of each component:

Table 1 HPC SDK Release Components

	Linux_x86_64		Linux_aarch64	
	CUDA 11.8	CUDA 12.3	CUDA 11.8	CUDA 12.3
nvc++	24.3		24.3	
nvc	24.3		24.3	
nvfortran	24.3		24.3	
nvcc	11.8.89	12.3.101	11.8.89	12.3.101
NCCL	2.18.5	2.18.5	2.19.3	2.19.3
NVSHMEM	2.10.1	2.10.1	N/A	2.10.1
cuBLAS	11.11.4.17	12.3.4.1	11.11.3.6	12.3.4.1
cuBLASmp	0.1.2	0.1.2	0.1.2	0.1.2
cuFFT	10.9.0.58	11.0.12.1	10.9.0.58	11.0.12.1
cuFFTMp	11.0.14	11.0.14	N/A	11.0.14
cuRAND	10.3.0.86	10.3.4.101	10.3.0.86	10.3.4.101
cuSOLVER	11.4.1.48	11.5.4.101	11.4.1.48	11.5.4.101
cuSOLVERmp	0.4.3	0.4.3	0.4.3	0.4.3
cuSPARSE	11.7.5.86	12.2.0.103	11.7.5.86	12.2.0.103
cuTENSOR	2.0.1	2.0.1	2.0.1	2.0.1
Nsight Compute	2023.3.1		2023.3.1	
Nsight Systems	2024.4.1		2024.4.1	
OpenMPI	3.1.5		3.1.5	
HPC-X	2.14	2.17.1	2.14	2.17.1
OpenBLAS	0.3.23		0.3.23	

	Linux_x86_64		Linux_aarch64	
	CUDA 11.8	CUDA 12.3	CUDA 11.8	CUDA 12.3
Scalapack	2.2.0			2.2.0
Thrust	1.15.1	2.2.0	1.15.1	2.2.0
CUB	1.15.1	2.2.0	1.15.1	2.2.0
libcu++	1.8.1	2.2.0	1.8.1	2.2.0

Chapter 3.

SUPPORTED PLATFORMS

3.1. Platform Requirements for the HPC SDK

Table 2 HPC SDK Platform Requirements

Architecture	Linux Distributions	Minimum gcc/ glibc Toolchain	Minimum CUDA Driver
x86_64	RHEL/CentOS 7.3 - 7.9 RHEL/CentOS/Rocky 8.0 - 8.7 RHEL 9.2 Fedora 33, 34 OpenSUSE Leap 15.2 - 15.4 SLES 15SP2, 15SP3, 15SP4 Ubuntu 18.04, 20.04, 22.04 Debian 10	C99: 4.8 C11: 4.9 C++03: 4.8 C++11: 4.9 C++14: 5.1 C++17: 7.1 C++20: 10.1	450.36.06
aarch64	RHEL/CentOS/Rocky 8.0 - 8.7 Rocky 9.2 Ubuntu 20.04, 22.04 SLES 15SP2, 15SP3, 15SP4 Amazon Linux 2	C99: 4.8 C11: 4.9 C++03: 4.8 C++11: 4.9 C++14: 5.1 C++17: 7.1 C++20: 10.1	450.36.06

Programs generated by the HPC Compilers for x86_64 processors require a minimum of AVX instructions, which includes Sandy Bridge and newer CPUs from Intel, as well as Bulldozer and newer CPUs from AMD. The HPC SDK includes support for v8.1+

Server Class Arm CPUs that meet the requirements appendix E specified in the SBSA 7.1 specification.

The HPC Compilers are compatible with gcc and g++ and use the GCC C and C++ libraries; the minimum compatible versions of GCC are listed in Table 2. The minimum system requirements for CUDA and NVIDIA Math Library requirements are available in the [NVIDIA CUDA Toolkit documentation](#).

3.2. Supported CUDA Toolchain Versions

The NVIDIA HPC SDK uses elements of the CUDA toolchain when building programs for execution with NVIDIA GPUs. Every HPC SDK installation package puts the required CUDA components into an installation directory called `[install-prefix]/[arch]/[nvhpc-version]/cuda`.

An NVIDIA CUDA GPU device driver must be installed on a system with a GPU before you can run a program compiled for the GPU on that system. The NVIDIA HPC SDK does not contain CUDA drivers. You must download and install the appropriate [CUDA driver from NVIDIA](#), including the [CUDA Compatibility Platform](#) if that is required.

The `nvaccelinfo` tool prints the CUDA Driver version in its output. You can use it to find out which version of the CUDA Driver is installed on your system.

The NVIDIA HPC SDK 24.3 includes the following CUDA toolchain versions:

- ▶ CUDA 11.8
- ▶ CUDA 12.3

The minimum required CUDA driver versions are listed in the table in Section 3.1.

Chapter 4.

KNOWN LIMITATIONS

- ▶ NVPL FFT respects the original thread affinity mask. For applications built with OpenMP runtimes, controls of thread affinity (either via `OMP_PROC_BIND` or `OMP_PLACES`) could negatively impact the multi-threaded performance. Users are recommended to unset the two OpenMP environment variables or set `OMP_PROC_BIND` to false for better performance. Additional known issues related to NVPL are described in the [NVPL documentation](#).
- ▶ The MPI wrappers in `comm_libs/mpi/bin` automatically detect the CUDA driver and select the matching MPI library from `comm_libs/X.Y`. Applications that require a full MPI directory hierarchy (e.g., `bin`, `include`, `lib`) or are launched via `srun` should bypass the MPI wrappers by loading the `nvhpc-hpcx-cuda11` or the `nvhpc-hpcx-cuda12` environment module, depending on the installed CUDA driver version.
- ▶ Passing an internal procedure as an actual argument to a Fortran subprogram is supported by `nvfortran` provided that the dummy argument is declared as an interface block or as a procedure dummy argument. `nvfortran` does not support internal procedures as actual arguments to dummy arguments declared external.
- ▶ `nvfortran` only supports the Fortran 2003 standard maximum of 7 dimensions for arrays (Fortran 2008 raised the standard maximum dimensions to 15). This limit is defined in the standard `CFI_MAX_RANK` macro in the `ISO_Fortran_binding.h` C header file.
- ▶ Section “15.5.2.4 Ordinary dummy variables”, constraint C1540 and Note 5 in the Fortran 2018 Standard allow Fortran compilers to avoid copy-in/copy-out argument passing provided that the actual and corresponding dummy arguments have the `ASYNCHRONOUS/VOLATILE` attribute, and the dummy arguments do not have the `VALUE` attribute. This feature is fully supported in `nvfortran` with `BIND(C)` interfaces (i.e., Fortran calling C). Copy-in/copy-out avoidance with asynchronous/volatile attributes may not be available in other cases with `nvfortran`.
- ▶ Some applications may see failures on Haswell and Broadwell with MKL version 2023.1.0 when running certain workloads with 4 or more OpenMP threads. The issue is resolved in MKL version 2023.2.0.
- ▶ `cuSolverMp` has two new dependencies on UCC and UCX libraries in the HPC-X directory. To execute a program linked against `cuSolverMP`, please use the “`nvhpc-hpcx-cuda11`” environment module for the HPC-X library, or set the environment variable `LD_LIBRARY_PATH` as follows: `LD_LIBRARY_PATH=`

```

${NVHPCSDK_HOME}/comm_libs/11.8/hpcx/latest/ucc/lib:${NVHPCSDK_HOME}/
comm_libs/11.8/hpcx/latest/ucx/lib:$LD_LIBRARY_PATH

```

- ▶ To use HPC-X, please use the provided environment module files or take care to source the `hpcx-init.sh` script: `$. ${NVHPCSDK_HOME}/comm_libs/X.Y/hpcx/latest/hpcx-init.sh`. Then, run the `hpcx_load` function defined by this script: `hpcx_load`. These actions will set important environment variables that are needed when running HPC-X. The following warning from HPC-X while running an MPI job – “WARNING: Open MPI tried to bind a process but failed. This is a warning only; your job will continue, though performance may be degraded” – is a known issue, and may be suppressed as follows: `export OMPI_MCA_hwloc_base_binding_policy=""`
- ▶ As of version 2.17.1, HPC-X does not have performance-optimal support for stream-ordered CUDA-allocated memory. In practical terms it means that IPC methods such as the MPI calls `MPI_Send` and `MPI_Recv` can have significantly degraded throughput when passed data allocated with the `cudaMallocAsync` function or its variants. This limitation will be removed in one of the future releases of HPC-X.
- ▶ Fortran derived type objects with zero-size derived type allocatable components that are used in sourced allocation or allocatable assignment may result in a runtime segmentation violation.
- ▶ When using `-stdpar` to accelerate C++ parallel algorithms, the algorithm calls cannot include virtual function calls or function calls through a function pointer, cannot use C++ exceptions, and must use random access iterators (raw pointers as iterators work best). When unified memory is not enabled, the algorithm calls can only dereference pointers that point to the heap. See the [C++ parallel algorithms documentation](#) for more details.

Chapter 5.

DEPRECATIONS AND CHANGES

- ▶ Support for the Power CPU architecture in the HPC SDK has been discontinued.
- ▶ The `-Mllvm` command line option for `nvc++`, `nvc`, and `nvfortran` is no longer supported.
- ▶ The GNU extension macros `linux` and `unix` are no longer defined when in ANSI mode (e.g., `-std=c++17` or `-std=c99`). If your code is compiled in ANSI mode and you rely on either of these macros, you will need to use one of the ANSI compliant macros `__linux__` or `__unix__`.
- ▶ Arm (aarch64) only: The 23.9 version of `nvfortran` changes the calling/return sequence for Fortran complex functions to match GNU's `gfortran` convention. Prior to the 23.9 release, `nvfortran` functions returned complex values via the stack using a "hidden" pointer as the first parameter. Now, complex values are returned following the `gfortran` convention via the floating-point registers. All libraries released with NVIDIA HPC SDK for Arm have been updated to follow the "gfortran" method. Users linking against Arm's performance libraries will need to use the "gcc" version instead of the "arm" version. All Fortran code, including libraries, that uses complex numbers must be recompiled when using `nvfortran` on Arm systems.
- ▶ Support for CUDA Fortran textures is deprecated in CUDA 11.0 and 11.8, and has been removed from CUDA 12. The 23.9 release is the last version of the HPC Compilers to include support for CUDA Fortran texture.
- ▶ The OpenMPI 3 library will be removed from the HPC SDK in a future release.
- ▶ The `-Minfo=intensity` option is no longer supported.
- ▶ The `CUDA_HOME` environment variable is ignored by the HPC Compilers. It is replaced by `NVHPC_CUDA_HOME`.
- ▶ The `-Mipa` option has been disabled starting with the 23.3 version of the HPC Compilers.
- ▶ The `-ta=tesla`, `-Mcuda`, `-Mcudalib` options for the HPC Compilers have been deprecated.
- ▶ Support for the RHEL 7-based operating systems will be removed in the HPC SDK version 24.9, corresponding with the upstream end-of-life (EOL).
- ▶ Starting with version 23.11, the HPC SDK bundles only CUDA 11.8 and the latest version of the CUDA 12.x series. Codepaths in the HPC Compilers that support CUDA versions older than 11.0 are no longer tested or maintained.

- ▶ Support for the Ubuntu 18.04 operating system has been removed in the HPC SDK version 23.5, corresponding with the upstream end-of-life (EOL).
- ▶ `cudaDeviceSynchronize()` in CUDA Fortran has been deprecated, and support has been removed from device code. It is still supported in host code.
- ▶ Starting with the 21.5 version of the NVIDIA HPC SDK, the `-cuda` option for NVC++ and NVFORTRAN no longer automatically links the NVIDIA GPU math libraries. Please refer to the `-cudalib` option.
- ▶ HPC Compiler support for the Kepler architecture of NVIDIA GPUs was deprecated starting with the 21.3 version of the NVIDIA HPC SDK.

Notice

ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE.

Information furnished is believed to be accurate and reliable. However, NVIDIA Corporation assumes no responsibility for the consequences of use of such information or for any infringement of patents or other rights of third parties that may result from its use. No license is granted by implication of otherwise under any patent rights of NVIDIA Corporation. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all other information previously supplied. NVIDIA Corporation products are not authorized as critical components in life support devices or systems without express written approval of NVIDIA Corporation.

Trademarks

NVIDIA, the NVIDIA logo, CUDA, CUDA-X, GPUDirect, HPC SDK, NGC, NVIDIA Volta, NVIDIA DGX, NVIDIA Nsight, NVLink, NVSwitch, and Tesla are trademarks and/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

© 2013-2024 NVIDIA Corporation. All rights reserved.