HPC SDK Release Notes

Release 25.7

NVIDIA Corporation

Jul 21, 2025

Contents

1 Release Component Versions **3**

2 Supported Platforms **5**
2.1 Platform Requirements for the HPC SDK 5
2.2 Supported CUDA Toolchain Versions 6

3 Known Limitations **7**

4 Deprecations and Changes **9**

NVIDIA HPC SDK Release Notes

Welcome to version 25.7 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 25.7 release of the HPC SDK includes component updates as well as important functionality and performance improvements.

- When using `-gcc-toolchain` flag during an application build, RPATH includes `/usr/lib64`
- NVIDIA Performance Libraries (NVPL) for Arm architectures, currently in beta, are included in the HPC SDK. NVPL is a collection of essential mathematical libraries optimized for Arm architectures that enable easy porting to NVIDIA GraceTM CPU platforms.
- Fortran interfaces are now available for cuBLAS GEAM routines
- The HPC SDK 25.7 ships with support for CUDA 12.9U1 and CUDA 11.8.

Chapter 1. Release Component Versions

The NVIDIA HPC SDK 25.7 release contains the following versions of each component:

Table 1: Table 1. HPC SDK Release Components

	Linux_x86_64		**Linux_aarch64**	
	CUDA 11.8	CUDA 12.9U1	CUDA 11.8	CUDA 12.9U1
nvc++	25.5		25.5	
nvc	25.5		25.5	
nvfortran	25.5		25.5	
nvcc	11.8.89	12.9.37	11.8.89	12.9.37
NCCL	2.18.5	2.26.5	2.19.3	2.26.5
NVSHMEM	3.2.5	3.2.5	N/A	3.2.5
cuBLAS	11.11.4.17	12.9.0.13	11.11.3.6	12.9.0.13
cuBLASMp	0.2.1	0.4.0	0.2.1	0.4.0
cuFFT	10.9.0.58	11.4.0.6	10.9.0.58	11.4.0.6
cuFFTMp*	11.2.6	11.4.0	N/A	11.4.0
cuRAND	10.3.0.86	10.3.10.19	10.3.0.86	10.3.10.19
cuSOLVER	11.4.1.48	11.7.4.40	11.4.1.48	11.7.4.40
cuSOLVERMp*	0.5.1.0	0.6.0.0	0.5.1.0	0.6.0.0
cuSPARSE	11.7.5.86	12.5.9.5	11.7.5.86	12.5.9.5
cuTENSOR	2.2.0	2.2.0	2.2.0	2.2.0
Nsight Compute	2025.2.0		2025.2.0	
Nsight Systems	2025.3.1		2025.3.1	
HPC-X	2.14	2.22.1	2.14	2.22.1
OpenBLAS	0.3.23		0.3.23	
Scalapack	2.2.0		2.2.0	
Thrust	1.15.1	2.8.2	1.15.1	2.8.2
CUB	1.15.1	2.8.2	1.15.1	2.8.2
libcu++	1.8.1	2.8.2	1.8.1	2.8.2
NVPL*	N/A		25.5	

* product in beta

Chapter 2. Supported Platforms

2.1. Platform Requirements for the HPC SDK

Table 1: Table 2. HPC SDK Platform Requirements

Architecture	Linux Distributions	Minimum gcc/glibc Toolchain	Minimum CUDA Driver
x86_64	RHEL/CentOS/Rocky 8.0 - 8.10 RHEL/Rocky 9.2 - 9.4 OpenSUSE Leap 15.4 - 15.4 SLES 15SP3, 15SP4, 15SP5, 15SP6 Ubuntu 20.04, 22.04, 24.04 Debian 12	Fortran, C, and up to C++17: 7.5 C++20: 10.1 C++23: 12.1	450.36.06
aarch64	RHEL/CentOS/Rocky 8.0 - 8.10 Rocky 9.2 - 9.3 Ubuntu 20.04, 22.04, 24.04 SLES 15SP6 Amazon Linux 2023	Fortran, C, and up to C++17: 7.5 C++20: 10.1 C++23: 12.1	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 the table in Section 3. The minimum system requirements for CUDA and NVIDIA Math Library requirements are available in the NVIDIA CUDA Toolkit documentation.

2.2. 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 25.7 includes the following CUDA toolchain versions:

- CUDA 11.8
- CUDA 12.9U1

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

Chapter 3. Known Limitations

The following are recommendations for more effectively using the HPC SDK and its components when unexpected behavior or suboptimal performance is encountered.

- HPC Compilers
  - When using nvfortran with `-g` and mixing Blackwell and non-Blackwell compute capabilities in the same fat binary, `-gpu=nodebug` is implied. Users can specify only Blackwell support with `-gpu=cc100` or `-gpu=120` when `-g` support on the device is needed.
  - For nvfortran, the `IOSTAT` argument of defined input/output procedures is expected to be of default kind `INTEGER`. `IOSTAT` declared to be other than the default kind may experience undefined behavior at runtime.
  - When a pointer is assigned to an array dummy argument with the target attribute, nvfortran may associate the pointer with a copy of the array argument instead of the actual argument.
  - 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.
  - 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.
- MPI, HPC-X, and UCX
  - The HPC SDK 25.7 ships with HPC-X version 2.22.1 which is incompatible with CUDA 12.0 driver (R525). HPC-X 2.20 is available as a fallback for users requiring CUDA 12.0. HPC-X

2.20 can be selected by loading the `nvhpc-hpcx-2.20-cuda12` environment module. The HPC-X UCC component can be re-enabled by setting `OMPI_MCA_coll_ucc_enable=1` environment variable.

- Any program data specified in `acc declare create` (and related clauses such as `copyin`, `device_resident`) can cause an application crash if used in an HPC-X MPI transport.
- 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.
- 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=""`
- Starting with 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 a future release.

- Math Libraries
  - Known issues related to NVPL are described in the NVPL documentation.
  - 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 dependencies on UCC and UCX libraries in the HPC-X directory. To execute a program linked against cuSolverMP using CUDA 11.8, 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`

Chapter 4. Deprecations and Changes

- Support for FMA4 and Piledriver class CPUs has been deprecated in the HPC Compilers.
- `CUDA_VISIBLE_DEVICES` is not supported at compile time when using nvfortran to generate GPU device code. The `-gpu=ccXY` option can be used to specify the desired code generation on systems with multiple GPU architectures.
- The Maxwell, Pascal, and Volta architectures are deprecated with CUDA 12.8, and support will be removed in a future version.
- Support for using stdpar with C++14 and below has been deprecated; C++17 or higher is required when using stdpar.
- When the next major version of CUDA becomes available, the HPC SDK will ship with the newest version of CUDA and CUDA 12.<latest>; CUDA 11.8 will be removed from the HPC SDK package and support will be deprecated.
- The `-M[no]-acle-intrinsics` option is a no-op beginning with version 25.1, and has been removed starting with 25.3.
- The nvvp and nvprof utilities have been deprecated and will be removed from a future release of the HPC SDK. Users of nvvp and nvprof are recommended to use the NSight Systems and NSight Compute applications.
- The OpenMPI 3 library has been removed from the HPC SDK starting with the 24.11 release. The OpenMPI 4 library will be removed in a future release.
- Support for CUDA versions 11.0 and 11.1 has been removed from the HPC SDK starting with the 24.11 release.
- The following flags have been deprecated and should not be used: `-Mllvm`, `-gpu=stacklimit`; `-gpu=pinned`, `-gpu=[no]managed`, `-gpu=[no]unified` (see the HPC Compilers User's Guide for more information).
- The following deprecated flags have been removed from the HPC Compilers starting with version 24.11:
  - `-Mcuda`, replaced by `-cuda`
  - `-Mcudalib`, replaced by `-cudalib`
  - `-ta`, replaced by `-acc=gpu`
- In a future release, the HPC SDK tar package file name will be extended to include the release number, in addition to the version. Automations that download and install the HPC SDK from the tar file package may need to be updated.
- Starting with version 24.7, the HPC compilers will not perform reciprocal rewrites at optimization level `-O3` or below; reciprocal rewrites are enabled with the `-Mfprelaxed` or `-Ofast` options.

- As of HPC SDK version 24.7 for Arm, UCC collectives were disabled by default for the HPC-X package. Users wishing to re-enable UCC collective operations can set `OMPI_MCA_coll_ucc_enable=1` in their environment. Performance on some systems may depend on whether UCC collectives are enabled or not.
- The effect of the `OMP_NUM_TEAMS` environment variable was changed in 24.7. It now specifies an upper bound on the number of teams, in accordance with the OpenMP specification. In previous releases, the number of teams was always set to `OMP_NUM_TEAMS`; now the value is decided by the OpenMP runtime and will be no greater than `OMP_NUM_TEAMS`. The `NVCOMPILER_OMP_CUDA_GRID` environment variable may be used to force a specific number of teams.
- Support for the Power CPU architecture in the HPC SDK has been discontinued.
- Support for the Amazon Linux 2 and RHEL 7-based operating systems was discontinued in the HPC SDK starting with version 24.9, corresponding with the upstream end-of-life (EOL).
- 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 gfortan 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 was the last version of the HPC Compilers to include support for CUDA Fortran `texture`.
- 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.
- 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.
- 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.

Notices

Notice and Disclaimers

All information provided in this document is provided as-is, for your informational purposes only and is subject to change at any time without notice. Reproduction of information in this document is permissible only if approved in advance by NVIDIA in writing. To obtain the latest information, please contact your NVIDIA representative. Product or service performance varies by use, configuration and other factors. Your costs and results may vary. No product or component is absolutely secure. TO THE FULLEST EXTENT PERMITTED BY APPLICABLE LAW, NVIDIA DISCLAIMS ALL WARRANTIES AND REPRESENTATIONS OF ANY KIND, WHETHER EXPRESS, IMPLIED OR STATUTORY, RELATING TO OR ARISING UNDER THIS DOCUMENT, INCLUDING, WITHOUT LIMITATION, THE WARRANTIES OF TITLE, NONINFRINGEMENT, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, USAGE OF TRADE AND COURSE OF DEALING. NVIDIA products are not intended or authorized for use as critical components in a system or application where the use of or failure of such system or application developed with products, technology, software or services provided by NVIDIA could result in injury, death or catastrophic damage.

Except for your permitted use of the information contained in this document, no license or right is granted by implication, estoppel or otherwise. If this document directly includes or links to third-party websites, products, services or information, please consult other sources to evaluate if and how to use that information since NVIDIA does not support, endorse or assume any responsibility for any third party offerings or its accuracy or usefulness.

TO THE FULLEST EXTENT PERMITTED BY APPLICABLE LAW, IN NO EVENT WILL NVIDIA BE LIABLE FOR ANY (I) INDIRECT, PUNITIVE, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES, OR (II) DAMAGES FOR THE (A) COST OF PROCURING SUBSTITUTE GOODS OR (B) LOSS OF PROFITS, REVENUES, USE, DATA OR GOODWILL ARISING OUT OF OR RELATED TO THIS DOCUMENT, WHETHER BASED ON BREACH OF CONTRACT, TORT (INCLUDING NEGLIGENCE), STRICT LIABILITY, OR OTHERWISE, AND EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES AND EVEN IF A PARTY'S REMEDIES FAIL THEIR ESSENTIAL PURPOSE. ADDITIONALLY, TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, NVIDIA'S TOTAL CUMULATIVE AGGREGATE LIABILITY FOR ANY AND ALL LIABILITIES, OBLIGATIONS OR CLAIMS ARISING OUT OF OR RELATED TO THIS DOCUMENT WILL NOT EXCEED FIVE U.S. DOLLARS (US$5).

Statements in this document that refer to future plans or expectations are forward-looking statements. These statements are based on currently available information, beliefs, assumptions and involve many risks and uncertainties that could cause actual results to differ materially from those expressed or implied in these statements. For more information on the factors that could cause actual results to differ materially, see our most recent earnings release and SEC filings at NVIDIA Corporation SEC Filings.

Trademarks

Copyright