NVIDIA HPC SDK RELEASE NOTES

RN-09976-001-V24.11 | November 2024

TABLE OF CONTENTS

LIST OF TABLES

Chapter 1. WHAT'S NEW

Welcome to version 24.11 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 24.11 release of the HPC SDK includes component updates as well as important functionality and performance improvements.

- The HPC Compilers support the C++ Standard Library feature std::atomic_ref on both CPU and GPU for 64-bit and 32-bit data types.
- An experimental backend based on the OpenACC runtime is available for a subset of the C++ parallel algorithms, and it can be enabled using the -stdpar=gpu:acc option to NVC++. Further information is available in the documenation.
- The HPC SDK version 24.11 ships with CUDA 12.6 update 2 and CUDA 11.8. Notable enhancements available in CUDA 12.6u2 include:
  - cuBLAS: Broad performance improvements on Hopper GPUs for FP8, BF16, and FP16 matmuls and fused epilogues.
  - cuSOLVER: A non-symmetric eigensolver and batched symmetric eigensolver are now available.
  - cuFFT LTO: cuFFT LTO callbacks are now generally available with additional performance gains.

Chapter 2. RELEASE COMPONENT VERSIONS

The NVIDIA HPC SDK 24.11 release contains the following versions of each component:

Table 1 HPC SDK Release Components

	Linux_x86_64		Linux_aarch64	
	CUDA 11.8	**CUDA 12.6**	**CUDA 11.8**	**CUDA 12.6**
nvc++	24.11		24.11	
nvc	24.11		24.11	
nvfortran	24.11		24.11	
nvcc	11.8.89	12.6.77	11.8.89	12.6.77
NCCL	2.18.5	2.18.5	2.19.3	2.19.3
NVSHMEM	3.0.7	3.0.7	N/A	3.0.7
cuBLAS	11.11.4.17	12.6.3.3	11.11.3.6	12.6.3.3
cuBLASMp	0.2.1	0.2.1	0.2.1	0.2.1
cuFFT	10.9.0.58	11.3.0.4	10.9.0.58	11.3.0.4
cuFFTMp	11.2.6	11.2.6	N/A	11.2.6
cuRAND	10.3.0.86	10.3.7.77	10.3.0.86	10.3.7.77
cuSOLVER	11.4.1.48	11.7.1.2	11.4.1.48	11.7.1.2
cuSOLVERMp	0.5.1.0	0.5.1.0	0.5.1.0	0.5.1.0
cuSPARSE	11.7.5.86	12.5.4.2	11.7.5.86	12.5.4.2
cuTENSOR	2.0.2	2.0.2	2.0.2	2.0.2
Nsight Compute	2024.3.2		2024.3.2	
Nsight Systems	2024.6.1		2024.6.1	
HPC-X	2.14	2.20	2.14	2.20
OpenBLAS	0.3.23		0.3.23	
Scalapack	2.2.0			2.2.0

	Linux_x86_64		Linux_aarch64	
	CUDA 11.8	CUDA 12.6	CUDA 11.8	CUDA 12.6
Thrust	1.15.1	2.5.0	1.15.1	2.5.0
CUB	1.15.1	2.5.0	1.15.1	2.5.0
libcu++	1.8.1	2.5.0	1.8.1	2.5.0

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/Rocky 8.0 - 8.10 RHEL/Rocky 9.2 - 9.4 OpenSUSE Leap 15.4 - 15.4 SLES 15SP3, 15SP4, 15SP5, 15SP6 Ubuntu 18.04, 20.04, 22.04, 24.04 Debian 10-12	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.

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 24.11 includes the following CUDA toolchain versions:

- CUDA 11.8
- CUDA 12.6u2

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

Chapter 4. KNOWN LIMITATIONS

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

- HPC Compilers
  - 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
  - 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.
  - A small number of applications have been found to either deadlock or segfault in the HPC-X UCC implementation of MPI_Allreduce and MPI_Reduce. The HPC-X UCC component can be disabled by setting OMPI_MCA_coll_ucc_enable=0 environment variable.
  - HPC-X users may notice longer startup overhead on MPI jobs that run for a very short period of time. The environment variable UCX_VFS_ENABLE=n can be set as a possible workaround.
  - On some systems UCX can fail while parsing /proc/self/map information, resulting in the crash with the following trace: "sys.c:194 UCX FATAL failed to allocate maps buffer(size=32768)" To work around this issue, set UCX_MEM_EVENTS=no environment variable.
  - 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 5. DEPRECATIONS AND CHANGES

- 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 here for more info).
- 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. When using UCC with HPC-X on Arm, users are advised to check the accuracy of their calculations.
- 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.

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.