NVIDIA Trusted Computing Solutions

Release Notes

RN-11468-001_r570 | February 2025

Document History

RN-11468-001_r570_02

Version	Date	Authors	Description of Change
01	December 2024		Initial release.
02	February 2025		R570 GA release

Table of Contents

Overview

This release consists of the NVIDIA® CUDA® Toolkit version 12.8, which is paired with the NVIDIA Data Center GPU Drivers version 570.86.15.

The following features are supported in this software release:

- The Protected PCIe (PPCIe) mode and Single GPU Passthrough (SPT) for NVIDIA H100 GPUs.
- Key Rotation in the SPT mode.

Refer to Feature Summary for more information about the supported Confidential Computing (CC) modes for H100 GPUs.

Before you deploy workloads, NVIDIA recommends that users use good practices, such as performing regular attestations.

Feature Summary

Confidential Computing

This section provides information about the CC features in this release.

Hopper Single GPU Passthrough with a Bounce Buffer

NVIDIA® Trusted Computing support for NVIDIA Hopper™ GPUs was first introduced with the Hopper Single GPU Passthrough with a Bounce Buffer (SPT CC) mode. In this mode, one GPU can be passed through for each Confidential VM (CVM). A bounce buffer stages encrypted data transfers between the GPU device and CVM. Refer to the *Intel TDX - Confidential Computing Deployment Guide and AMD SNP - Confidential Computing Deployment Guide* for more information.

Table 1. Component Versions to Enable the SPT CC Mode

Component	Version
VBIOS	v96.00.5E.xx.xx.xx or later.
CVM Kernel	• Intel TDX Kernel 6.9 • AMD SEV 5.19
gpu_admin.py	The main branch is github.com/nvidia/nvtrust.
Attestation/Verifier	Version 1.4.0 or later.

Protected PCIe

This section provides information about the PPCIe features in this release.

Eight Hopper GPUs with Four NVSwitch Passthrough

Trusted Computing support in the PPCIe mode is available **only** with the Hopper GPUs and Intel® CPUs with TDX technology in an Ubuntu KVM/QEMU environment.

In the PPCIe mode, multiple NVSwitch/NVLink interconnected Hopper GPUs can be passed through to one CVM. As in the SPT CC mode, a bounce buffer is used to stage encrypted data transfers between the GPU device and CVM over the PCI Express bus. In this mode, GPU-GPU communications over the NVLink/NVSwitch are not encrypted (refer to the *Protected PCIe Deployment Guide* PDF file, which is a part of this posting, for more information).

Table 2. Component Versions to Enable PPCIe

Component	Version
HGX firmware bundle	1.6.0
CVM Kernel	Intel TDX Kernel 6.9 **Note**: AMD systems have not yet been validated.
gpu_admin.py	The main branch is github.com/nvidia/nvtrust.
Attestation/Verifier	2.1.0

Limitations

This section provides a list of the known limitations in this release.

Limitations in the Hopper SPT CC Mode

- Only one GPU per CVM is allowed.
  - Only one CVM is permitted even in systems with multiple GPUs.
  - This limitation is temporary and is expected to be resolved in a future release.
- With a maximum of one GPU passed through per CVM, operations that involve multiple GPUs, such as P2P communications, are not supported.

Limitations in the Hopper PPCIe Mode

- Hopper PPCIe is limited to HGX 8-way systems, where the eight GPUs and four NVSwitches are passed through to one VM.
  Other topologies are not supported.
- NVIDIA NCCL is the only supported GPU communication library.
- In the PPCIe mode, when the source or destination operand are imported, GPU memory allocations on a device that is not visible to the process, the host-to-device, or device-to-host copies might fail asynchronously with `cudaErrorLaunchFailure`.
- In the PPCIe mode, using `cooperative_groups::multi_grid_group::sync` in kernels launched with `cudaLaunchCooperativeKernelMultiDevice` results in the kernel failing with `cudaErrorIllegalAddress`.
- CUDA Interprocess Communication (IPC) is not supported in PPCIe mode.
- Developer tools such as NVIDIA Nsight for profiling are not supported in PPCIe mode

Limitations in the SPT CC and PPCIe Modes

This section provides information about the limitations that apply to the SPT CC and PPCIe modes.

The following runtime APIs are incompatible with CC:

- Host memory registration.
  The following CPU memory pinning operations are not allowed in CC mode:
  - cudaHostRegister
  - cudaHostUnregister
- cudaMemcpy calls that describe an HtoA or AtoH copy.

The following Host-to-Array and Array-to-Host copies are not supported because of the potential requirement for a conversion between pitch-linear and block-linear access patterns of the CUArray memory type during the secure copy operation:
    - cudaMemcpy2DFromArray
    - cudaMemcpy2DFromArrayAsync
    - cudaMemcpy2DToArray
    - cudaMemcpy2DToArrayAsync
    - cudaMemcpy3D
    - cudaMemcpy3DAsync
    - cudaMemcpy3DPeer
- CUDA External Resource Interoperability.
  The following APIs are not supported because an external resource interaction with a trusted execution environment is not permitted:
    - cudaImportExternalMemory
    - cudaExternalMemoryGetMappedBuffer
    - cudaExternalMemoryGetMappedMipmappedArray
    - cudaDestroyExternalMemory
    - cudaFreeMipmappedArray
    - cudaImportExternalSemaphore
    - cudaSignalExternalSemaphoresAsync
    - cudaWaitExternalSemaphoresAsync
    - cudaDestroyExternalSemaphore
    - cudaGraphAddExternalSemaphoresSignalNode
    - cudaGraphAddExternalSemaphoresWaitNode
    - cudaGraphExecExternalSemaphoresSignalNodeSetParams
    - cudaGraphExecExternalSemaphoresWaitNodeSetParams
    - cudaGraphExternalSemaphoresSignalNodeGetParams
    - cudaGraphExternalSemaphoresSignalNodeSetParams
    - cudaGraphExternalSemaphoresWaitNodeGetParams
    - cudaGraphExternalSemaphoresWaitNodeSetParams

The following Driver APIs are incompatible with CC:

- Host memory registration.
  The following CPU memory pinning operations are not allowed in CC mode:
    - cuMemHostRegister
    - cuMemHostUnregister
- cuMemcpy calls that describe an HtoA or AtoH copy.
  The following Host-to-Array and Array-to-Host copies are not supported because of the potential requirement for a conversion between pitch-linear and block-linear access patterns of the CUArray memory type during the secure copy operation:
    - cuMemcpy2DUnaligned

    - cuMemcpyAtoH
    - cuMemcpyAtoHAsync
    - cuMemcpyHtoA
    - cuMemcpyHtoAAsync
- cuStream memory operation calls passing pointers allocated using cudaMallocHost, cudaHostAlloc, cuMemAllocHost APIs, and their graph counterparts:
    - cuStreamBatchMemOp
    - cuStreamBatchMemOp_v2
    - cuStreamWaitValue32
    - cuStreamWaitValue32_v2
    - cuStreamWaitValue64
    - cuStreamWaitValue64_v2
    - cuStreamWriteValue32
    - cuStreamWriteValue32_v2
    - cuStreamWriteValue64
    - cuStreamWriteValue64_v2
    - cuGraphAddBatchMemOpNode
    - cuGraphBatchMemOpNodeGetParams
    - cuGraphBatchMemOpNodeSetParams
    - CuGraphExecBatchMemOpNodeSetParams
- CUDA External Resource Interoperability.
  The following APIs are not supported as external resource interaction with a trusted execution environment is not permitted:
    - cuImportExternalMemory
    - cuExternalMemoryGetMappedBuffer
    - cuExternalMemoryGetMappedMipmappedArray
    - cuDestroyExternalMemory
    - cuFreeMipmappedArray
    - cuImportExternalSemaphore
    - cuSignalExternalSemaphoresAsync
    - cuWaitExternalSemaphoresAsync
    - cuDestroyExternalSemaphore
    - cuGraphAddExternalSemaphoresSignalNode
    - cuGraphAddExternalSemaphoresWaitNode
    - cuGraphExecExternalSemaphoresSignalNodeSetParams
    - cuGraphExecExternalSemaphoresWaitNodeSetParams
    - cuGraphExternalSemaphoresSignalNodeGetParams
    - cuGraphExternalSemaphoresSignalNodeSetParams
    - cuGraphExternalSemaphoresWaitNodeGetParams
    - cuGraphExternalSemaphoresWaitNodeSetParams

The following CUDA capabilities are incompatible with CC:

- CUDA/Graphics interop, specifically APIs to enable interop with EGL, VDPAU, OpenGL, DirectX, OptiX, and Vulkan.
- GPUDirect RDMA.
- The CUDA Programmatic Dependent Launch and Synchronization feature will not show expected overlaps in the primary and secondary kernel executions.
  A program that uses these APIs should functionally succeed in CC modes.

The following CUDA samples are expected to fail when you run them in the CC mode:

- convolutionTexture
- dct8x8
- lineOfSight
- simpleCubemapTexture
- simpleLayeredTexture
- simplePitchLinearTexture
- simpleStream
- simpleTexture
- simpleTextureDrv
- watershedSegmentationNPP

The following CUDA samples are expected to fail in the PPCIe mode:

- simpleIPC
- cudaCompressibleMemory
- p2pBandwithLatencyTest

The following CUDA Runtime APIs are not supported with CC in this release but might be enabled in a future release:

- cudaEventElapsedTime
- cudaEventCreateWithFlags where Flags is set to cudaEventBlockingSync

The following CUDA capabilities are not supported with CC in this release but might be enabled in a future release:

- CUDA Multi Process Service (MPS).
- CUDA Toolkit minor version compatibility.
- CUDA Forward Compatibility.

Known Issues

- A key rotation feature is not supported with PPCIe.
  A sophisticated attacker with physical or logical superuser access to the system can act as a passive adversary to capture the ciphertext and execute an attempt to break it or the key.

  Workaround
  Users should review the latest research on the effects of extreme AES key usage and the cryptographic wear out to determine their requirements for an attacker advantage. To create a new set of encryption keys in PPCIe mode, users must terminate and launch their CVMs again.

- IV exhaustion will crash the application in PPCIe mode.
  The H100 CC modes use a 96-bit deterministic IV for each virtual copy engine that is used to transfer data between the GPU and CPU. When this IV space is exhausted, transfers will fail to complete.

  Workaround
  Rotate the keys often in supported modes. If the keys are not rotated often, restart the CVM.

- GPU-Ready bit is set when the devtools mode is enabled.

  Workaround
  When in full CC-on modes, the driver will not accept any workloads until after the Attestation SDK, or the users, manually enable a GPU-Ready bit.

Note This bit is already enabled in the Devtools mode.

Users should use best practices by attesting the GPU before performing any work. The GPUs booted in devtools mode will be clearly identified, and the attestation will fail.

- With HGX Firmware 1.6.0, there is an increased risk of GPU/NVSwitch falling off the PCIe bus during DC power cycling. This will be resolved in a future firmware release.

  Workaround
  A system reboot would need to be performed to bring the missing devices back on the PCIe bus

- NVIDIA Performance Primitives might not work.

  NVIDIA Performance Primitives (NPP) uses optimized coding to extract the maximum performance from commonly used transforms/calculations as part of the leverage pinned host memory, which is not supported in CC.

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.

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.

No license, either expressed or implied, is granted under any NVIDIA patent right, copyright, or other NVIDIA intellectual property right under this document. Information published by NVIDIA regarding third-party products or services does not constitute a license from NVIDIA to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property rights of the third party, or a license from NVIDIA under the patents or other intellectual property rights of NVIDIA.

Reproduction of information in this document is permissible only if approved in advance by NVIDIA in writing, reproduced without alteration and in full compliance with all applicable export laws and regulations, and accompanied by all associated conditions, limitations, and notices.

THIS DOCUMENT AND 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. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL NVIDIA BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA’s aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms of Sale for the product.

Trademarks

NVIDIA, the NVIDIA logo 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.

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

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