



Confidential Compute on NVIDIA Hopper H100

Document History

WP-11459-001

| Version | Date | Authors | Description of Change |
|---------|---------------|-------------|--------------------------------|
| 1.0 | July 25, 2023 | Rob Nertney | Initial Release – Early Access |

Contents

- Introduction.....5
- NVIDIA GPUs have long since been used as an extremely effective method to accelerate the processing of enormous amounts of data, from deep learning to high-performance computing. The NVIDIA H100 GPU is NVIDIA’s flagship product which now includes confidential computing enablement.6
- Scope and Audience of this Document6
- Security - Why Should You Care? A Study of Privacy Preserving Technologies.....6
- A Description of Secure Systems7
 - What is a Trustworthy System?.....7
 - Chain of Trust – Hardware Validity8
 - Confidential Computing – A Feature for Secured Systems 10
 - How NVIDIA H100 Integrates into a TVM’s TEE 10
 - Secure and Trusted Boot..... 11
- Confidential Computing Features of the NVIDIA Hopper H100..... 12
 - Goals for Hopper H100 Confidential Computing 13
 - Threats and Mitigations..... 14
 - In-Scope Threat Vectors 15
 - Out-of-Scope Threat Vectors 15
 - Confidentiality..... 15
 - In-Band Attacks 15
 - Integrity..... 16
 - Modifying a Payload..... 16
 - Replaying a Message 17
 - Side-Channel Attacks..... 18
 - Performance Counters..... 18
 - Availability 18
- Setting up a Confidential Compute Environment with H100..... 19
 - Steps 1-4 - Defining Host Topology and Configuring GPUs for CC Mode.....20
 - Steps 5-7 - CVM Attests the GPU and Begins Executing Code21
 - Secure Session Establishment.....21
 - Attesting the GPU.....22
- Running a Confidential Compute Application on the GPU26

| | |
|-------------------------------|----|
| Dataflow in CC Modes | 26 |
| Developer Considerations..... | 27 |
| Performance Samples | 28 |
| Summary | 31 |

Disclaimer

NVIDIA will provide Early Access releases of the software solution outlined in this paper. These early solutions may not support all of the claims made in this paper. Until General Access is released NVIDIA highly recommends developers to utilize these releases for representative data and code only.

Introduction

Information is becoming increasingly valuable: Rapid digital transformation has led to an explosion of sensitive data being generated, stored, transferred and processed from datacenter on-premises, in the cloud or at the edge.

By harnessing the power of AI, this wealth of data presents an opportunity for enterprises to extract actionable insights, unlock new revenue streams, and improve customer experience to gain a competitive edge in today's data-driven business landscape.

As the world enters the exascale-era of data collection and data brokering, ensuring control of your data is paramount. Couple this with the distributed nature of how the data is generated, transferred, stored and used across multiple platforms and deployment models, the ability to retain confidence in data and model owner to have its control is tested. Many government regulatory regimes are being imposed across markets—across continents—requiring a certain level of data protection that prevents access and use of private data. Complex and evolving nature of the data privacy laws and regulations such as General Data Protection Regulation (GDPR) in Europe, Health Insurance Portability and Accountability Act (HIPAA), and Gramm-Leach-Bliley Act (GLBA) in the United States can pose significant challenges to organizations to gain the value of AI use from using those sensitive data.

Until recently, securing data was limited to data-in-motion (for example, moving a payload across the internet with SSL/TLS), and data-at-rest (e.g., encryption of storage media). Data-in-use, however, remained at risk. With Confidential Computing, users are able to close this gap and to enable more holistic end-to-end data protection model.

Enter Confidential Computing (CC): the ability to process data and code in-use securely, preventing unauthorized users from both access and modification. It allows users to have end-to-end data protection model and strengthen the layered security approach. The primary governing body of CC is the Confidential Computing Consortium (C3), a project at the Linux Foundation; C3 is comprised of the top hardware and software vendors in the world, all of whom have the unified goal of standardizing Confidential Compute across both hardware and software.

The steps of enabling confidential compute are the following:

- > Secure the hardware interfaces to the CPU or GPU (for example, DRAM & PCIe).

- > Secure the software from the owner/operators of the systems from unauthorized access to end-user data in what is called a Trusted Execution Environment (“TEE”).

C3 set forth in defining standards for attesting the hardware’s validity, setting up secure sessions, to and from that hardware, while simultaneously working with Hypervisor vendors to provide mechanisms to end-users that would ensure their data security, even from a ‘super user’ with direct access to the memory.

NVIDIA GPUs have long since been used as an extremely effective method to accelerate the processing of enormous amounts of data, from deep learning to high-performance computing. The NVIDIA H100 GPU is NVIDIA’s flagship product which now includes confidential computing enablement.

Scope and Audience of this Document

This document is designed to provide a high-level overview of the CC capabilities of the NVIDIA H100 GPU. Specific details on industry standards such as encryption/authentication algorithms, certifications may be beyond the scope of this document.

Other documents may be found at <https://docs.nvidia.com/confidential-computing>

Security - Why Should You Care? A Study of Privacy Preserving Technologies

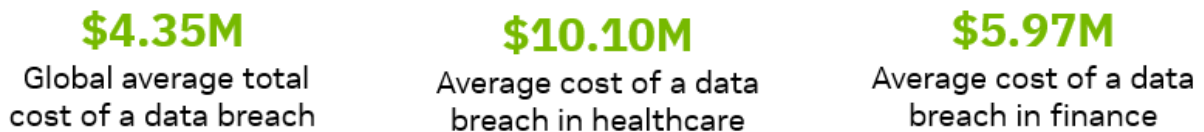
Broad market agreement with many technology consultancies such as [Gartner](#), [Deloitte](#), [McKinsey](#), [BCG](#), or [Capgemini](#), International Data Corporation (IDC) highlights the need for broader data collaboration and sharing to extend competitive advantages in AI and data centric solution development. For many market verticals natural market competitiveness, regulatory and privacy concerns create barriers to collaboration (even within the same organization) diluting value of data within the organization and increasing market friction for cross company/industry data collaborations. Even government organizations like DHS/US VA, US Census, and other large sources of data are challenged to realize their full potential due to regulatory and privacy barriers within the public and public/private spheres.

Increasingly, organizations are looking to privacy enhancing technologies (PETs) to help navigate the complex regulatory, privacy, and market barriers to realize the full potential of the local and shared data opportunities. As noted by the analysts, PETs are increasingly a driving force of innovation and value creation within data ecosystems.

The CC Consortium defines CC as the following:

“Confidential Computing protects data in use by performing computation in a hardware-based attested Trusted Execution Environment. These secure and isolated environments prevent unauthorized access or modification of applications and data while in use, thereby increasing the security assurances for organizations that manage sensitive and regulated data. Today, data is often encrypted at rest in storage and in transit across the network, but not while in use in memory. Additionally, the ability to protect data and code while it is in use is limited in conventional computing infrastructure. Organizations that handle sensitive data such as Personally Identifiable Information (PII), financial data, or health information need to mitigate threats that target the confidentiality and integrity of either the application or the data in memory.”

Figure 1 Cost of Data Breaches ¹



Source: [IBM Cost of Data Breaches report](#)

A Description of Secure Systems

This section provides information about secure systems.

What is a Trustworthy System?

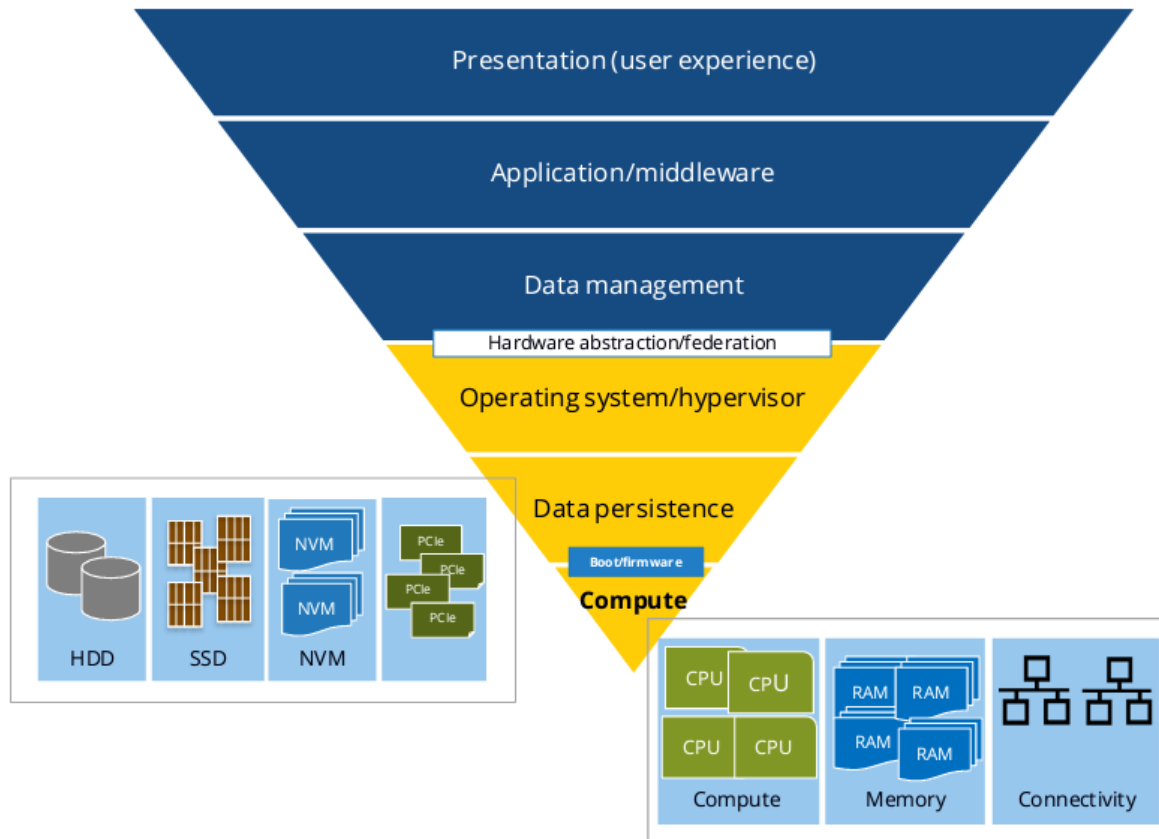
CC Consortium is helping drive the industry towards a standardized flow to ensure confidentiality. However, confidentiality in a system is a multilayered stack: to achieve the highest level of trust, you must start with the base hardware of the system (Figure 2). It may seem enough to begin with physical security of the nodes in a datacenter: restricted entry, vetted technicians, and isolated management networks, but securing a system actually begins even before the system is powered on for the first time.

A trustworthy system is made up of hardware and software that can be verified as trustworthy using mechanisms to verify the authenticity of technologies that make up the system. In the context of using confidential computing, the users need to verify the trustworthiness of the devices and firmware before the user can run their workload while protecting the data and code in use.

Looking at the layers of technologies that make up systems, each layer relies on the technology in the subsequent layer in the infrastructure stack. To verify the trustworthiness of the system, it is important to start with verifying the trustworthiness of the lowest technology of the infrastructure stack, which is the hardware compute and work your way up.

Figure 2. A Secure System Stack

The Infrastructure Stack



Source: IDC, 2019

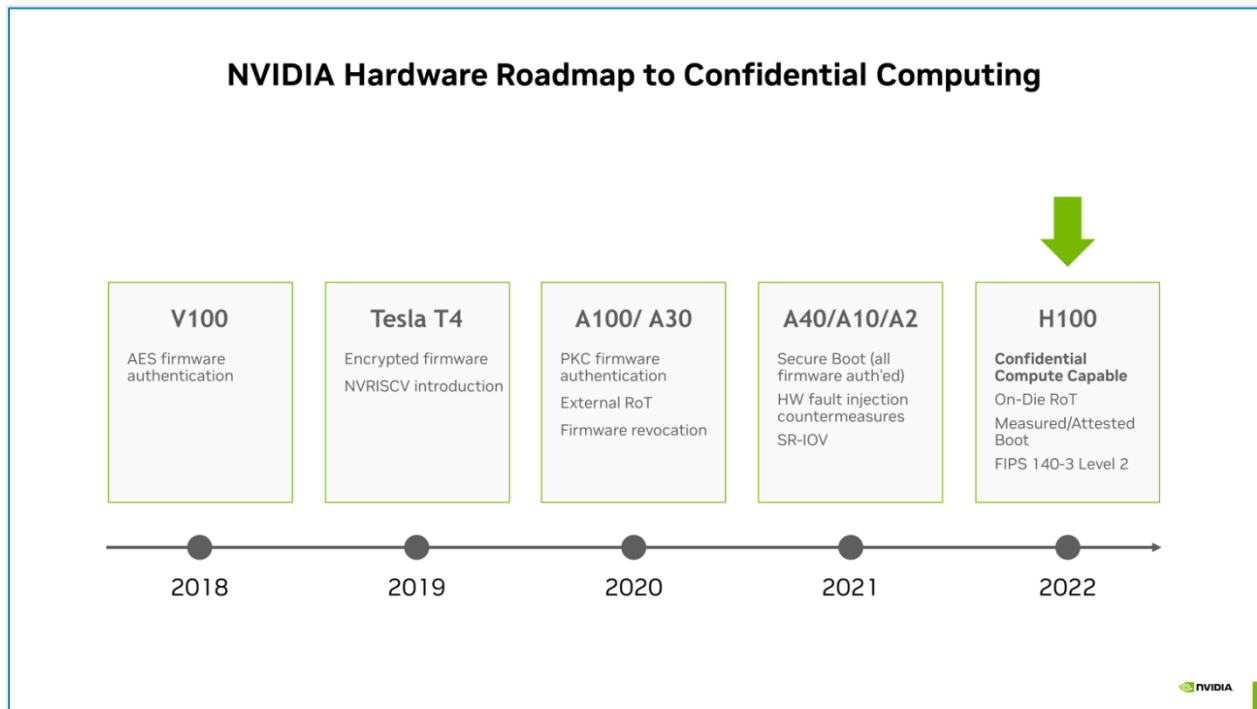
Chain of Trust – Hardware Validity

The *Chain of Trust* is a phrase to describe the process of validating the trustworthiness of a system from the end unit at every step until you reach the final authority, or “Root” that can be inherently trusted. In other words, a verifiable paper trail where at any point you can inspect and ensure the validity of a system’s trustworthiness. As a developer seeking confidential computing capabilities, you should be vigilant in scoping the trustworthiness of not just a single piece of a device or its firmware, but the overall system.

Many silicon providers have created their own methods of providing proof of authenticity. For example, the CPU vendor providing proof of silicon authenticity or restricting its use to a specific OEM, the motherboard vendors providing similar proof for their coprocessors (for example, a BMC) and the associated firmware-based components like the BIOS, UEFI, or BMC code. Paired together, owner-operators of standard compute nodes have built the foundation for Confidential Computing.

NVIDIA is no stranger to the world of validating the chain of trust:

Figure 3. The Road to Confidential Computing



For the last four generations, NVIDIA has been constantly improving the security and integrity of its devices. One of the first documented efforts was in the NVIDIA V100 GPU, where we provided AES Authentication on the firmware that runs on the device. This authentication enabled assurances that users could trust that the bootup firmware was neither corrupted, nor tampered with. As time went on, we introduced features, from encrypting the firmware in T4 so that malicious attackers could not easily review for potential security vulnerabilities, to the Ampere nodes, which added an external microcontroller review of the firmware (*ERoT*) to determine if it was still valid (*revocation*) and culminating with Hopper, a fully confidential computing capable GPU.

In a chain of trust, the trustworthiness of each layer of software that composes the chain is guaranteed by the previous layer, until reaching the root of the chain, or Root of trust (RoT). Immutability and formal verification provide the foundation for a root of trust. In CC, we use the On-Die RoT to verify the key fused onto the GPU to verify the trustworthiness of the GPU identity, and to also verify the trustworthiness of the firmware used.

Combined with the RoT, Secure Boot verifies that a GPU firmware has been signed by NVIDIA and only allows execution of the signed and authenticated firmware to be used during the GPU boot. It is a mechanism is used to authenticate and load the GPU firmware modules. Hardware mechanisms ensure the firmware cannot be modified subsequent to the load process until a subsequent reset. Combination of these two mechanisms ensure the GPU will always run only authenticated and uncorrupted firmware.

Confidential Computing is a *feature* of the secure hardware, and must rely on attested hardware before an expectation of confidentiality can occur. With hardware vendors

providing methods for you to ensure that what arrives on your datacenter's receiving docks, you have confidence that at no point in the (often global) shipping process was the hardware meaningfully modified without being able to detect it.

Confidential Computing – A Feature for Secured Systems

So far, the descriptions have been strictly ensuring validity of the hardware, and not yet focused specifically on Confidential Computing. These features are new, and constantly evolving to meet the performance demands of developers and confidentiality requirements set forth by organizational regimes, as such, you must have specific CPU hardware SKUs to enable Confidential Compute with NVIDIA's H100:

- > Intel CPUs must support “Trusted Domain eXtensions” (TDX)
- > AMD CPUs must support “Secure Encrypted Virtualization with Secure Nested Paging” (SEV-SNP)
- > ARM CPUs must support ARM “Confidential Compute Architecture” (CCA)

These CPUs have hardware-based features which are required for CC, where both vendors have similar methods for isolating virtual machines (VMs) that want to remain confidential (Confidential Virtual Machine (CVM)), with unique differences which achieve the same goal: isolation of a VM from outside sources.

The above hosts have hardware features to support confidential compute. Each of these hosts could use a combination of techniques like access control checks, paging control, address translation and memory encryption to provide protection of data while it's in use. See [Common Terminology for Confidential Computing](#) for more details.

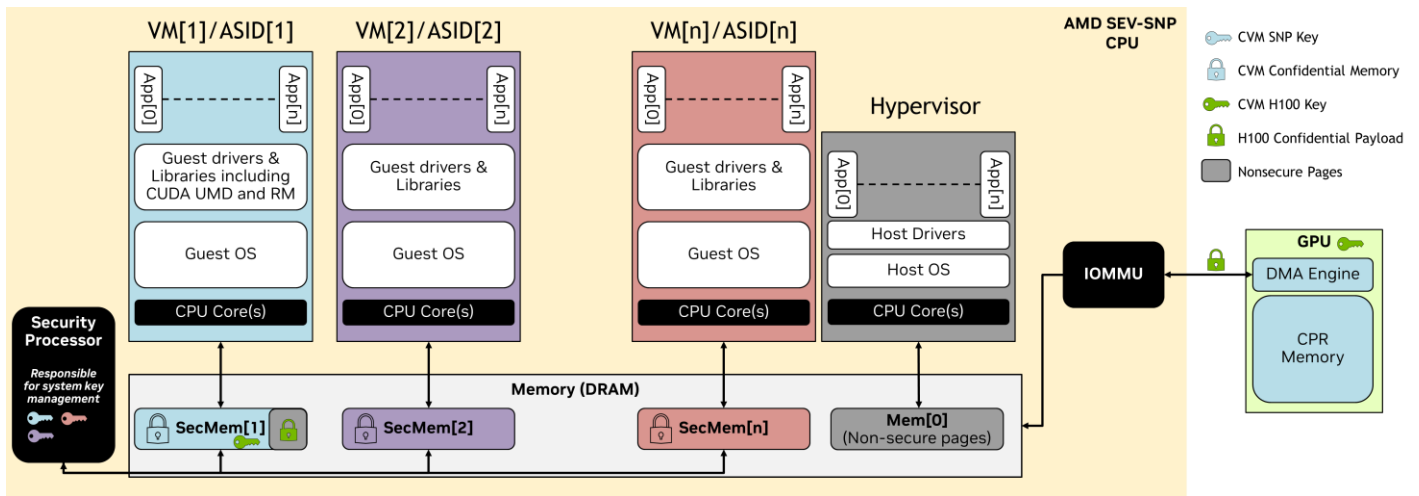
How NVIDIA H100 Integrates into a TVM's TEE

The first problem to solve is providing the GPU access to the encrypted data within the existing framework of how the CPUs are already handling memory pages in the system; the MMU in the processor is already configured to prevent unauthorized memory access between VMs, but the data is encrypted.

CPU Vendors have provisioned for the allocation of nonsecure pages while in confidential mode, which means that anyone in the system can gain access to it. However, as long as the data in these nonsecure pages remains encrypted, confidentiality is maintained.

These pages are visible to the GPU which can then transfer payloads into its internal storage. From a hardware perspective, the VMs are responsible for encrypting the data, sending it to this bounce buffer where the GPU can copy it to its internal memory, decrypt it, and process it. Once complete, the GPU will encrypt the data once more, place it back in the bounce buffer, and inform the VM that the data is ready for consumption.

Figure 4. Adding a GPU to an AMD SEV-SNP TEE



Similar to the CPU vendors, CC capabilities in the H100 requires many new and innovative hardware features. We'll be going over the specifics of these internal features, modes of operation, and how developers can start to use it, but there is one last topic within a secure system to consider: Secure Boot.

Secure and Trusted Boot

Once the system has arrived and been ingested into your datacenter, all of these features from the various hardware vendors are connected, and the process of enabling a Confidential system can truly begin. However, many activities begin long before the Operating System is ready to provision VMs with GPUs attached.

When the system powers on, many components will begin their aforementioned self-checks (their RoT; and in some cases, components may intercommunicate); should all the tests pass, the system begins the process of booting into a host operating system. In what is colloquially known as "Secure Boot", the motherboard is configured to store a set of Certificates in nonvolatile memory that can be used to authenticate the validity of software binaries that try to load for execution. Many OEMs will have the Certificates preloaded from the main OS vendors (for example, Canonical, Red Hat, and Microsoft). As the UEFI firmware fetches these OS bootloader binaries, their signatures are checked against the preloaded certificates. If the binary is unsigned or fails the check (e.g., for a signature without a matching Certificate, Revoked certificates, etc.), the system halts its boot process.

Assuming the first bootloader binary passes the signature/certificate check, the UEFI firmware will load and pass execution off to it; this authenticated bootloader can then load the final kernel. Since this kernel still has full access to the system firmware, it must also be validated against certificates in the system. Once loaded, these validated kernels will disable this access as it continues to boot.

The kernel drivers still have access to the boot firmware, and as such needs to ensure that I/O devices in the system are allowed to operate (again via checking the signatures; this time on the hardware drivers). It begins its own query of the hardware in the system, and via a handshake to the motherboard vendor, it requests whether the certificate is stored and valid and matches the signature of the drivers loading.

At any point, a failure can stop the boot process, or in the most recent case, disable the driver, but continue to boot.

As mentioned before, CC with NVIDIA H100 relies on a CPU with AMD SEV-SNP or Intel-TDX. These are virtual-machine based solutions and as such need to have an almost repeated step; at this point in the boot flow, the Host (owner/operator) OS has done a base validation of the hardware in the system but has implicitly trusted the hardware vendors that they “did the right thing”. An extra step that should be done is called “Attestation”, wherein the Host may request certain hardware to provide “evidence” that they “are who they say they are” and that they are running the correct levels of firmware, configured correctly, and so on.

Attestation of running hardware is paramount to the adage “trust but verify”. The Host should run attestation on all available hardware before launching their hypervisor and creating VMs for their final Confidential workloads. The Host, after doing their own attestation checks, will pass appropriate hardware to their hypervisor and will provision a CVM. Within this CVM, the kernel launching must again revalidate the signatures of any drivers attempting to gain direct access to system hardware as described above. Once the system has completed boot, the CVM is ready to be transferred to the end-user. They may (and should) utilize further attestation for the CVM within the infrastructure of the Host’s Owner-Operator (e.g., Azure or GCP Attestation services).

We will discuss how NVIDIA H100 GPUs should be provisioned and how it performs its attestation later in this document.

Confidential Computing Features of the NVIDIA Hopper H100

We’ve established a basic understanding of what makes a secure system, how the market has begun to leverage CC, and how an NVIDIA GPU can integrate into the existing flows. We’ve covered the hardware checking up through power-on, ensuring that the devices can do some initial cross-checks for supported/authorized hardware and ready-states. Once the CVM begins to load, the NVIDIA GPU Driver will begin to initiate many processes to prepare the GPU for use in a Confidential system.

To have a true production-ready confidential accelerator, NVIDIA needed to have robust firmware and software stacks with attestation flows to provide a complete CC solution which includes protection and integrity of both code and data. The Hopper H100 has

several new hardware-based features added to its silicon that enable this level of Confidentiality.

The Hopper Architecture is first brought to market in the H100 product, which includes GPU and HBM on a single package. There are multiple products using H100 including Vulcan (HGX), Viking (DGX), Scout (HPC), and H100 PCIe. These products are capable of supporting CC.

Goals for Hopper H100 Confidential Computing

There are three main goals NVIDIA set for enabling the H100 with CC, aligned to the charter of the CC Consortium:

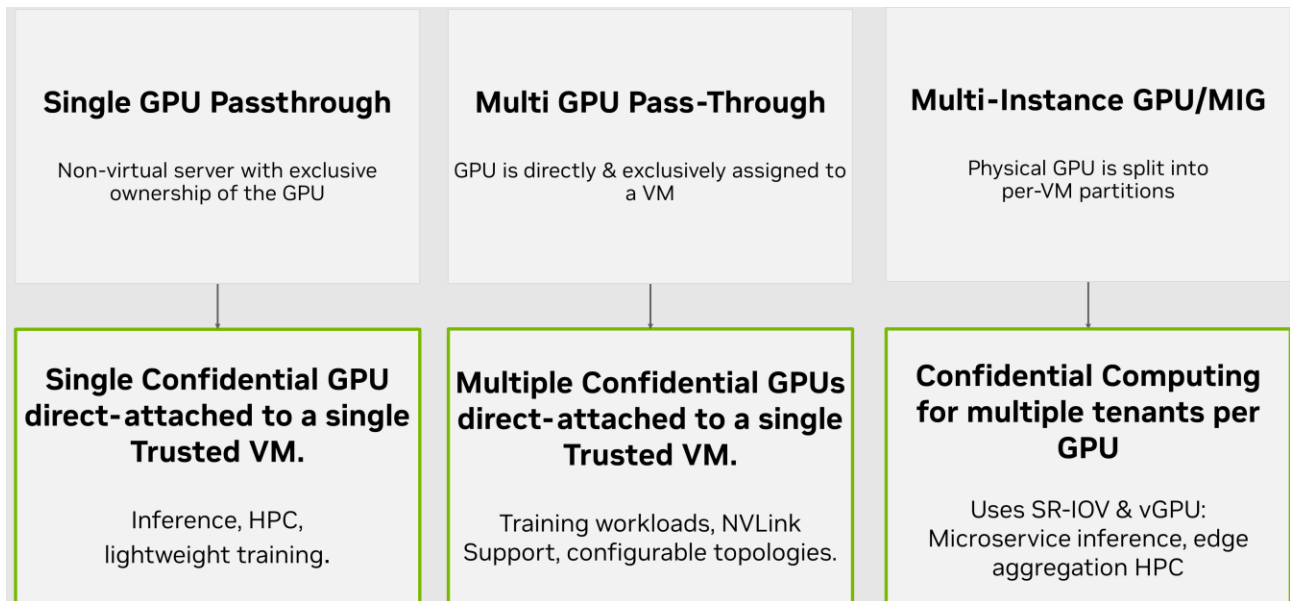
- > Data and Code Confidentiality
 - Protect all application code and data in the VM instance from being read by the host.
- > Data and Code Integrity
 - Protect all application code and data in the VM instance from being altered by the host.
- > Basic Physical Attacks
 - Interposers on buses such as PCIe and DDR memory cannot leak data or code.

Here are the main modes of delivering NVIDIA GPUs to a VM:

- > Assigning a GPU to a VM.
- > Passing multiple GPUs to a VM.
- > Passing a partial GPU to a VM using vGPU and SRIOV.

These deployment modes can also be directly mapped to CC modes (refer to Figure 5).

Figure 5. Usecases for H100 Confidential Computing



Every Hopper H100 devices will support the use-cases, and each use case supports the following operation modes:

- > CC-Off –
 - Standard H100 operation. None of the encryption/authentication paths are active, and none of the other CC-specific system blocks/firewalls/sideband-channels are active.
- > CC-On –
 - The H100, with the drivers on the CPU will have fully activated all the CC features available; all firewalls are active, and all sideband channels have been closed.
- > CC-DevTools –
 - Many developers count on our Developer Tools to help debug their code and profile it such that they can understand system bottlenecks to improve overall performance. In CC-Devtools mode, the GPU is in full “CC-On” mode, as described above, except for the data paths required to run the developer tools.

Threats and Mitigations

The goals for Hopper H100’s CC mode can be further subdivided into the categories outlined in Figure 6. We will go over each of the subcategories (Confidentiality, Integrity, Availability, and General) in turn, describing how the H100 mitigates a particular threat at a high level.





In-Scope Threat Vectors

- > Software attacks
- > Basic physical attacks
- > Software Roll back attacks
- > Cryptographical attacks
- > Data Roll-back & replay attacks

Out-of-Scope Threat Vectors

- > Sophisticated Physical attacks
- > Denial of Service attacks

Figure 6. Threats and Mitigations of H100's Confidential Computing Modes

| Category | Threat | Mitigation |
|--|--|------------|
|  Confidentiality | Use PCIe/NVLink to read tenant data (e.g. Hypervisor, another VM, PCIe interposer) | ✓ |
| | Use Out-of-band management/debug channels to read tenant data (e.g. SMBus, JTAG) | ✓ |
| | Use memory remapping to read tenant data | ✓ |
| | Use GPU Cache/Memory based side channels to read tenant data | ✓ |
| | Use GPU TLB based side channels to read tenant data | ✓ |
| | Use GPU Performance Counters to read tenant data or fingerprint tenant | ✓ |
|  Integrity | Read tenant data via hypothetical physical attacks (physical side channels / DPA / EM, HBM interposer) | ✗ |
| | Use PCIe/NVLink to modify tenant data (e.g. Hypervisor, another VM, PCIe interposer) | ✓ |
| | Use Out-of-band management/debug channels to modify tenant data (e.g. SMBus, JTAG) | ✓ |
| | Corrupt tenant data by replaying previous data or MMIO transactions (replay attacks) | ✓ |
|  Availability | Corrupt tenant data via hypothetical physical attacks (fault injection, HBM interposer) | ✗ |
| | Denial of Service to hypervisor by tenant | ✓ |
| | Denial of Service to tenant by another tenant | ✓ |
| | Permanent denial of service of GPU by tenant | ✓ |
|  General | Denial of Service to tenant by hypervisor | ✗ |
| | Use a spoofed, non-genuine, or known vulnerable TCB component | ✓ |
| | Use hardware side channels (e.g. DPA) to extract persistent device keys | ✓ |
| | Use hardware side channels (e.g. DPA) to extract tenant ephemeral session key | ✗ |

Confidentiality

This section provides information about confidentiality in CC.

In-Band Attacks

The confidentiality subcategory describes how your data is kept secret from malicious or otherwise unauthorized actors. There are many methods in which one may consider snooping or otherwise obtaining confidential code, from hardware-based snooping of the physical interfaces to software code attempting to access memory or GPUs not assigned to it; some of these mitigations are inherited from the CPU vendors' efforts to keep a traditional CVM secure, while others are novel from NVIDIA and specific to securing a GPU.

The first, and easiest to visualize is by simply using the physical PCIe or NVLink connections to read tenant data. PCIe bus analyzers are quite common throughout the industry, particularly when debugging new silicon or testing for PCI SIG compliance. Traditionally, these interposers are physically connected between the host and the target device, quietly recording all activity that goes across the copper lines of the connector, and then refactoring the data into human readable format. NVIDIA has built-in encryption and decryption engines across all its ingress/egress paths; these engines use 256-bit AES-GCM encryption. Any request or response which enters or exits the GPU must be encrypted.

However, without changing the key, sending the same plaintext message multiple times will result in the same encrypted data on the bus; this is unacceptable as it could provide an attacker with knowledge of the system, payloads, and so on. A solution of this is to introduce a further layer of protection with something called a changing the Initialization Vector (IV), which can be used to effectively randomize each message; the base key does not change, but rather mathematically modified by the IV, which changes with each payload. This benefit is twofold:

- > Multiple plain-text messages now produce entirely different ciphertexts with the same key.
- > Introduces a strongly ordered behavior to the payloads, which mitigates a commonly known attack vector called Replay Attacks.

In AES-GCM, a 96bit IV is required.

Even with this addition of IVs, AES-GCM is still somewhat vulnerable to reusing the (Key, IV) pair for encrypting different data blocks. The total number of “uniqueness” is limited to $(2^{96})-1$, after which an IV is considered “exhausted” and the encryption key must be rotated out and replaced.

Even with this addition of IVs, AES-GCM is still somewhat vulnerable to reusing the (Key, IV) pair for encrypting different data blocks. The total number of “uniqueness” is limited to $(2^{96})-1$, after which an IV is considered “exhausted” and the encryption key must be rotated out and replaced. The NVIDIA software stack automatically supports this rotation, with options for user-modification to the default policy.

All data being transferred out of the developers’ TVM or out of their provisioned GPU(s) will be encrypted in such a manner. The “bounce buffer” mentioned in “How NVIDIA H100 Integrates into a TVM’s TEE” on page 10 will only contain encrypted data. Data in the VM is stored in host memory and is encrypted and secured with facilities provided by the CPU vendors.

Integrity

Modifying a Payload

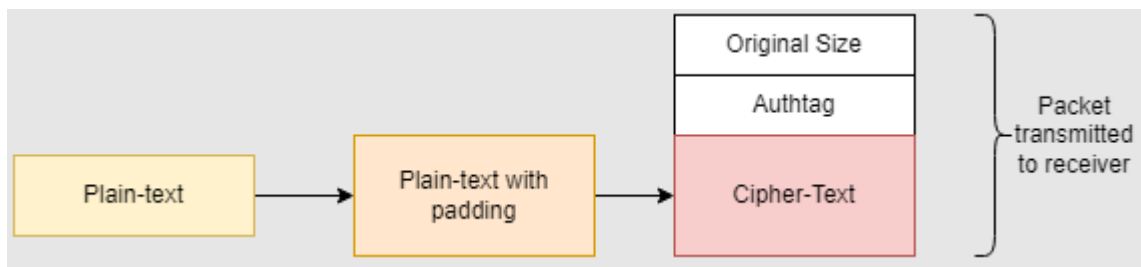
In a similar vein to the Confidentiality section, where we discussed the mitigations used to prevent an adversary from reading your data, however, modifying it in flight cannot be

mitigated by encryption alone. It is true that (especially with rolling IVs) it is highly improbable that modifying encrypted traffic could result in a determinable outcome, but it can absolutely introduce silent errors or discrepancies that may go entirely unnoticed.

The AES-GCM algorithm also provisions for the creation of a digital signature, called an AuthTag, along with the ciphertext. It uses its key to “sign” the ciphertext to create a unique fingerprint of the payload; this is a cryptographically secure method that is pragmatically improbable for an adversary to modify the ciphertext and/or the associated AuthTag.

Upon receiving the payload on the remote end, the receiver will perform the same operation using its copy of the key to calculate its own version of the AuthTag. Should they not match, the decryption will fail, and an error will be thrown.

Figure 7. The Secured Payload



Replaying a Message

The next logical place for an attacker to attempt to interfere with your code is something called a “Replay Attack”. This is an attack where a *man in the middle* intercepts an otherwise valid payload and at some point in the future sends it again. An analogy would be for an adversary to have identified a payload that transmitted q, “Password accepted, continue onward and trust me”. If that particular payload was captured, it could be replayed at an opportune moment giving the adversary access to the user’s password.

Again, the AES-GCM rolling IV feature on the H100 comes to mitigate this threat. Before, we mentioned how a rolling IV had a secondary feature to prevent Replay-Attacks. When both endpoints of the secure channel are related, each can keep a counter of messages as the IV increments. The receiving endpoint performing the decryption doesn’t use the IV sent by the encrypting transmitter, instead it uses its own.

Similar to the IV increment after encryption, the decrypting end-point will increment the IV after decryption. If an adversary or malicious user tries to capture and replay, both the decryption and AuthTag comparison will fail as IV has already been incremented.

Side-Channel Attacks

In many enterprise/hyperscale datacenters, where the number of nodes can reach thousands-to-millions, the need for remote management to access, upgrade, or debug hardware is paramount. Normally, the motherboards in these servers have a secondary light-weight coprocessor, called a Board Management Controller (BMC).

The BMC has its own network interface, memory, network, and so on, and through a System Management Bus (SMBus), and technicians can log into this separate BMC, and obtain full system access to everything in the node:

- > Virtual Desktop (like plugging in a local monitor and keyboard) to the main CPU complex.
- > Node hardware access & information (UEFI/BIOS updates, Temperature, Inventory, Power control, etc.)
- > Peripheral access & information (GPU inventory, firmware updates, temperature, performance counters, etc.)

This sideband is extremely powerful and can be catastrophic if access falls into the wrong hands. The H100 will lock out access to any paths which may access tenant data and reduce remaining paths to an anonymous 'card health' only level of detail.

JTAG (named after the Joint Test Action Group which codified it) access, commonly used as another method of debugging electronic components of any sort, is also disabled when the GPU is operating in Confidential Mode.

Performance Counters

When using NVIDIA Developer Tools (DevTools) to profile the performance/behavior of your application, special device-side hardware counters are utilized to help measure on-device code. However, performance counters could be used to infer behavior of the usage of the device and provide avenue for side channel attacks. These are disabled in full CC-On mode. However, since profiling is a critical to optimizing application performance, we support a development mode called CC-Dev Tools. While in CC-DevTools mode all of the encryption paths, bounce buffers, etc., are enabled but access to these counters are unblocked for developer use.

Availability

Availability involves the overall health of the system in relation to a bad actor attempting to deny or interfere with the operation and/or access of another tenant, or the hypervisor itself. The H100 and its software stacks do not require any special hooks into the Hypervisor, and as such, cannot interfere with the operation of the host. The traditional methods of isolation/protection of Hypervisor from the VM are supplemented with the new CPU features (TDX and SEV-SNP) designed to isolate the VM from a potentially malicious Hypervisor.

Traditionally, an H100 can only access memory locations owned by its assigned VM: addresses configured by the Hypervisor into the IOMMU. TDX and SEV-SNP go a step further and will only permit IOMMU-allowed transactions to access the memory pages marked as **Shared** within the CVM. This purpose of the combined security is as follows:

- > It provides the standard protections of multiple tenants from denying service to other tenants in the system (malicious H100 trying to access another VM, or malicious VM trying to access other valid VMs and/or their hardware).
- > It prevents a malicious H100 from compromising the CVM's memory space that it should not have access to.

NVIDIA has long been integrated into the top public Cloud Service Providers (CSPs) who provide semi-autonomous, direct access to hardware within their infrastructure; it is impractical to have a complete vetting of every potential user who may access these systems. Together, NVIDIA partners with these CSPs to ensure that any malicious user who utilizes their services cannot leave the GPU in an otherwise unusable state for the next tenant, either via host crashes, PCIe-bus lockups, or physically damaging the GPU or other parts of the infrastructure. Any permanent administrative access to the GPU is locked out of in-band control.

The only out of scope Availability attack is one where a malicious Hypervisor prevents access to a CVM; these attacks could include removing tenant network access, disconnecting the GPU power, etc. While these attacks cannot have software blocks, all of the private keys used to encrypt/decrypt the virtual machine and GPU memories are in keyslots which cannot be accessed by the Hypervisor. These keys are ephemeral and are lost/wiped on VM teardown or GPU reset.

Setting up a Confidential Compute Environment with H100

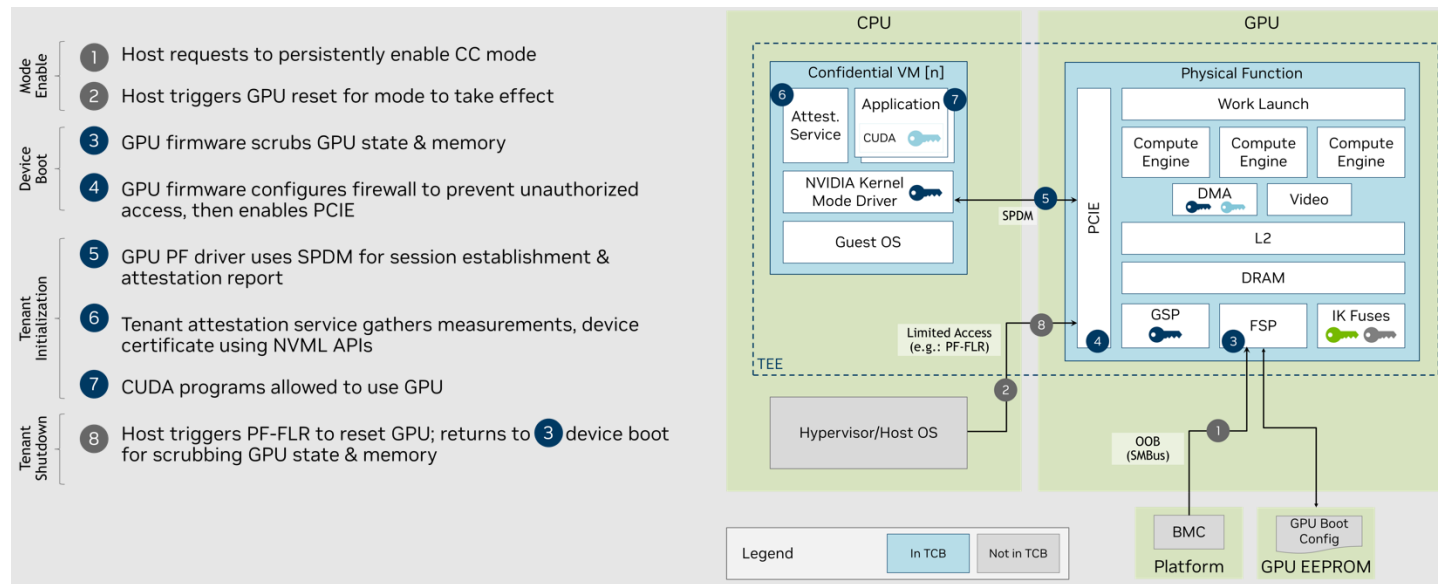
This next section will go over the details of how a CVM operates when using an H100. Specific details on the flows, supported versions, example code, etc., are covered in the *Deployment Guide for Trusted Environments*, which can be found on our [GitHub](#). The Deployment Guide will define multiple personas:

- **Hardware IT Administrator**
- **Host OS Administrator**
- **Virtual Machine Administrator**
- **Virtual Machine User**
- **Container User**

For the setup flows; as secured systems, especially those targeting CC, touch every level of the system stack, different groups will be responsible for and/or have different experiences, requirements, and steps needed to succeed in these environments.

The steps are outlined in Figure 8.

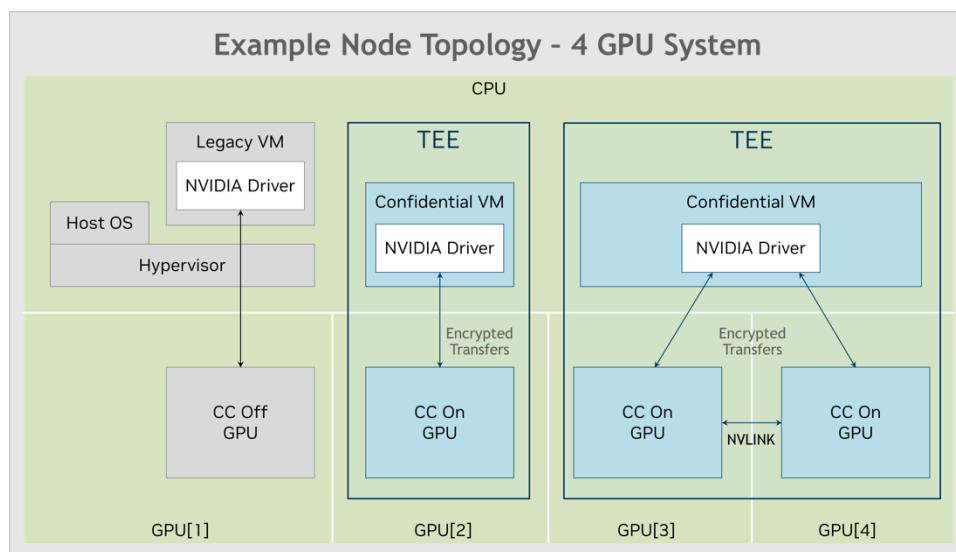
Figure 8. The H100 CC Initialization Process



Steps 1-4 - Defining Host Topology and Configuring GPUs for CC Mode

The Hopper H100 CC modes, as described in “Goals for Hopper H100 Confidential Computing” on page 13 must be configured by the hypervisor. The CPU features, along with the H100 software stacks, allow a combination configuration of these modes. Figure 9 illustrates an example.

Figure 9. Sample Node Topology using H100



In the figure above, there are 4 GPUs in the system. The hypervisor can dynamically change the Confidential modes on any of the GPUs as required by their provisioning.

The APIs to enable/disable CC are provided as both in-band PCIe commands from the Host or out-of-band BMC commands. A toggle operation requires a Function Level Reset (FLR) of the GPU for the mode to take effect. During this reset, a memory lock is engaged which blocks access to the GPU's memory until it has been scrubbed (mitigating cold boot attacks). GPU Firmware initiates a scrub of memory and states in registers and SRAMs before the GPU is handed over to the user. In addition, it also configures the GPU firewall to prevent unauthorized access to the GPU.

Steps 5-7 - CVM Attests the GPU and Begins Executing Code

This section provides additional information about steps 5-7.

Secure Session Establishment

During loading of the secure VM on the CPU, the NVIDIA GPU driver will be loaded. This triggers the VM's first communication with the device beyond basic PCIe enumeration.

A secure session must be created to allow for encryption and decryption of data moving between the VM and the GPU. This encrypted session is required for the following reasons:

- > The GPU cannot access the protected pages of the VM.
This means data flowing to and from the GPU must pass through unprotected bounce buffers where entities outside the TCB (for example, the hypervisor) could read/write them.
- > PCIe is not a secure or private channel between the VM and the tenant, data travelling over it may be read or corrupted by entities outside the TCB.

There are two parts to establishing the secure session:

- > A Diffie-Hellman key exchange for setting up a shared symmetric session key.
- > Retrieving the Device Attestation Report that contains the measurements of the HW and SW components.

All firmware and microcode running on the GPU is authored and signed by NVIDIA. This is verified before it executes. However, retrieving and verifying the attestation report is a critical step in building trust in the device. It verifies that the versions are what is expected and trusted by the user of the device. Once the attestation report has been verified by checking the signature against the NVIDIA Root CA and the symmetric key is established the device is almost ready to be used by CUDA applications running in the TEE. Trust has been established in the device and a secure channel has been created.

Attesting the GPU

Attestation is the process where users, or the *relying party* wants to challenge the GPU hardware and its associated driver, firmware, and microcode, and receive confirmation that the responses are both valid and authentic before proceeding.

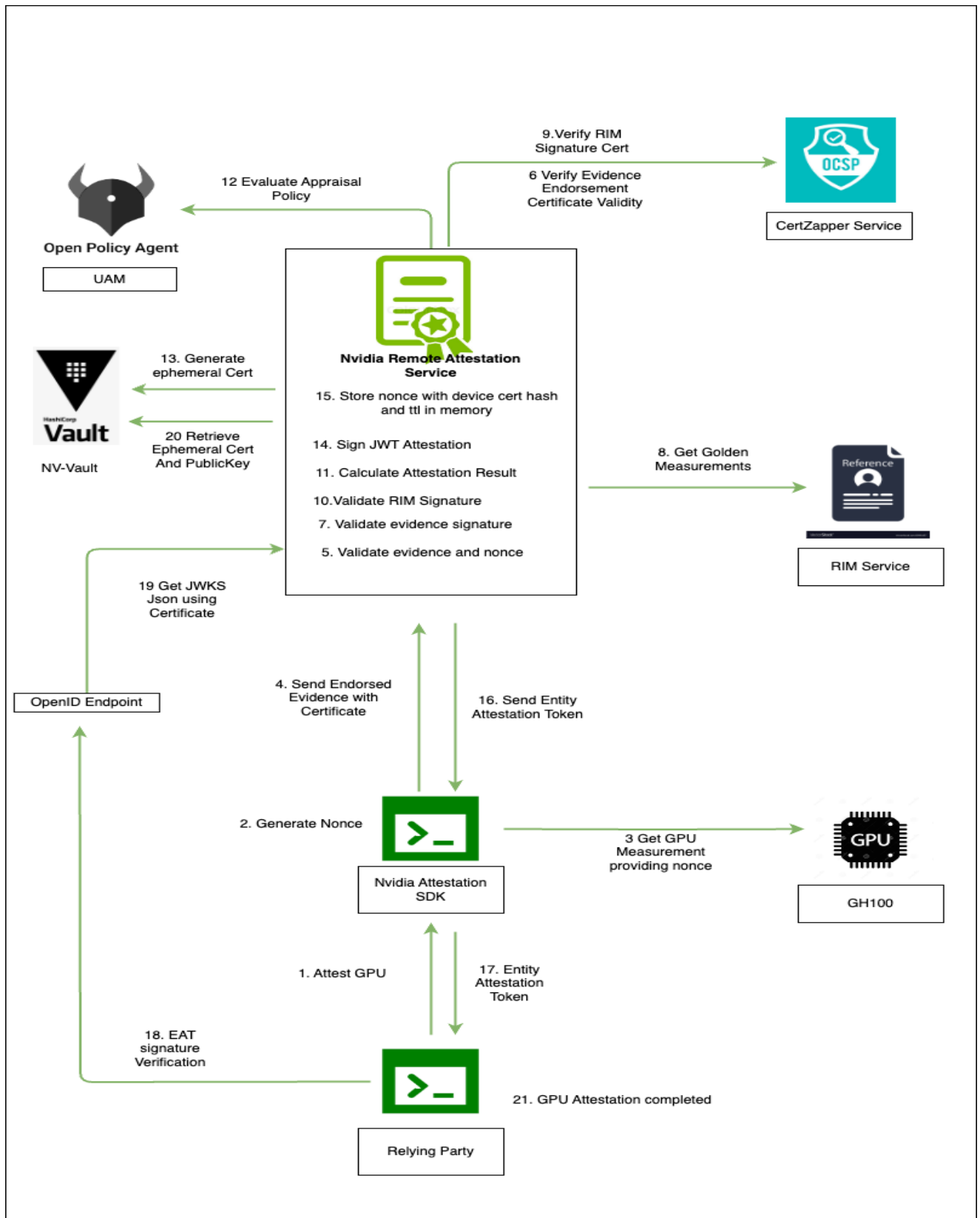
Before a CVM uses the GPU, it must authenticate all GPUs as genuine before including it in its trust boundary. It does this by retrieving a Device Identity Certificate (signed with a device-unique ECC-384 key pair) from the device or the NVIDIA Device Identity Service. It can be retrieved using the Per Device Identifier (PDI) that is embedded into every GPU Chip. Verification of this certificate against NVIDIA's Certificate Authority will verify the device was manufactured by NVIDIA. This device-unique private Identity Key (IK) is burned into fuses of each H100; the public key is retained, but all copies of these private keys are destroyed during the manufacturing process.

In addition, the TVM must also ensure that the GPU certificate is not revoked by comparing it with a Certificate Revocation List (CRL) that can be fetched by services like Online Certificate Service Protocol (OCSP).

While utilizing the NVIDIA Remote Attestation Service (NRAS) will be the primary method of validating the attestation reports, options will be provided to truly air-gapped scenarios to provide local verification (noting of course, that stale local data regarding revocation status or integrity of the verifier may occur).

Figure 10 illustrates the overall attestation flow for attestation.

Figure 10. The NVIDIA Remote Attestation Service



Here are the general steps for the Attestation workflow using the NVIDIA Remote Attestation Service:

1. The Relying party will use the NVIDIA Attestation SDK to attest GPU or the NRAS APIs.
2. The NVIDIA Attestation generates a random nonce of at least 128-bits using a secure random number generator meeting NIST SP800-90 A/B/C standards..
3. The NVIDIA Attestation SDK sends the nonce to the attester to get a measurement of the device (GPU) using the cc_admin/nvml library.
4. The attester embeds the nonce in endorsed evidence.
5. This nonce will also be passed in the payload to remote attestation.

The evidence will be signed using a private key called Attestation Key and the certificate containing the public key corresponding to the Attestation Key will be shared to verify the signature of the evidence. Evidence is endorsed with an Attestation Key (AK). The AK, for a GPU, is generated deterministically at each full chip reset. A certificate is generated for the AK and signed by a device identity key that is per device. That certificate chain has one or more intermediate certificates above the identity key to prove trust in the AK.

6. The SDK will send the endorsed evidence with the device identity and attestation key certificates and nonce to Remote Attestation Service on HTTPS (TLS session)
7. NRAS validates the request , parse the Evidence and validate against SPDM Specification.
8. NRAS also validates the nonce which is stored in memory for the Replay attack detection.
9. NRAS maintains the nonce with a hash of the device certificate public key and attach a TTL of 24 hours.
10. Additional requests with the same nonce for the same device will be rejected by the same server instance (pod) as the in-memory cache will have the same nonce already processed.
11. NRAS validates the evidence endorsement certificate chain.

If any of the certificates in the certificate chain is revoked, NRAS will return an an **Evidence is not valid.** message.

12. NRAS validates the signature of the evidence using the public key from the verified certificate chain (see RFC 5280) and check if the issuer of the certificate chain belongs to NVIDIA PKI.
13. If the signature of the evidence is not able to be verified, and an error will be generated and the attestation evaluation will be stopped.

HTTP Error codes 400 with the message Invalid Signature.

14. NRAS fetches RIM Bundle (Golden Measurements) from RIM Service for the device using the driver-version and GPU model provided in the evidence.

15. NRAS to RIM Service communication will be retried if there are any connection errors.

Any exception from the RIM service, for example, **RIM Bundle Not Found.** will result in GPU attestation failure and the **Unsupported Driver Version.** will be sent.

NRAS can cache the RIM bundles which are frequently accessed considering the same RIM Bundle won't get updated and changed.

16. NRAS gets the certificate chain from RIMM Bundle and call the CertZapper OCSP endpoint to validate whether the certificate is valid. If the certificate is revoked, NRAS will send an attestation failure with an error response as "RIM Bundle is revoked".

17. NRAS validates the signature of the RIM bundle against the public certificate.

18. NRAS compares the Evidence with the Golden measurement fetched from RIMM Service and create Attestation Result.

19. NRAS calls UAM (OPA rego policy engine) to apply appraisal policy for evidence. This is for scenarios where some of the measurements are not matched but it's known to the verifier and the verifier as an owner can still decide whether the attestation is good or not. There is also relying party appraisal policy which the relying party applies on the end (EAT) attestation result.

20. NRAS uses an ephemeral certificate L3 for signing the EAT.

a. This L3 certificate will be short-lived, 24 hours of ttl.

21. NRAS creates EAT (Entity Attestation Token) in JWT format for verified claims.

22. NRAS uses the ephemeral keys and Certificate and sign the JWT Token using private keys.

23. NRAS sends an attestation token (JWT) to the SDK.

24. JWT has the JWT Id to prevent replay attack (jti) and also an expiry time for which the result will be valid.

Any failure in the steps 11-14 will result in an Internal Server error and internal retries will be implemented after retries are exhausted, the error response with an **Internal Server Error, try again later.** message.

25. NRAS stores the nonce with device certificate hash with ttl of 24 hours in memory which is successfully processed before sending the attestation result to relying party. This data is needed to avoid replay attacks.

26. NRAS sends the result to Attestation SDK.

27. SDK sends back the response (EAT) to the Relying Party.

28. The relying Party validates the token using OpenID public keys provided either as part of JWT or as an issuer endpoint.

OpenID endpoint is the JWKS endpoint which will have the Public Certificate for validating the EAT, this endpoint is served by the NRAS Service.

29. NRAS retrieves the certificate from the Vault PKI engine.

30. On successful validation, the relying party will apply its own Appraisal policy to decide whether the device is in a good state.

Based on the decision, the relying party will allow the device to communicate or block it.

Running a Confidential Compute Application on the GPU

Once the CPU TEE's trust has been extended to the GPU, running Compute applications is identical to a regular GPU. NVIDIA has worked extensively to ensure that your code works. Once the steps in "Attesting the GPU" on page 22 have been completed with the correct hardware, drivers, and a passing attestation report, executing your kernels should be transparent.

Dataflow in CC Modes

After a CVM with the H100 has been correctly configured, booted, and attested to, you can start securely processing data on your H100 GPU. We worked to ensure as much of a *lift and shift* style of coding as possible. The goal is to have the existing code and kernels from users work without changes when H100 CC modes are enabled.

By default, devices are blocked from interacting with CVM and cannot directly access CVM memory. Below we describe how the driver enables H100 to securely communicate with the CVM in CC mode.

Figure 11. Confidential H100 GPU with an AMD SEV-SNP TEE

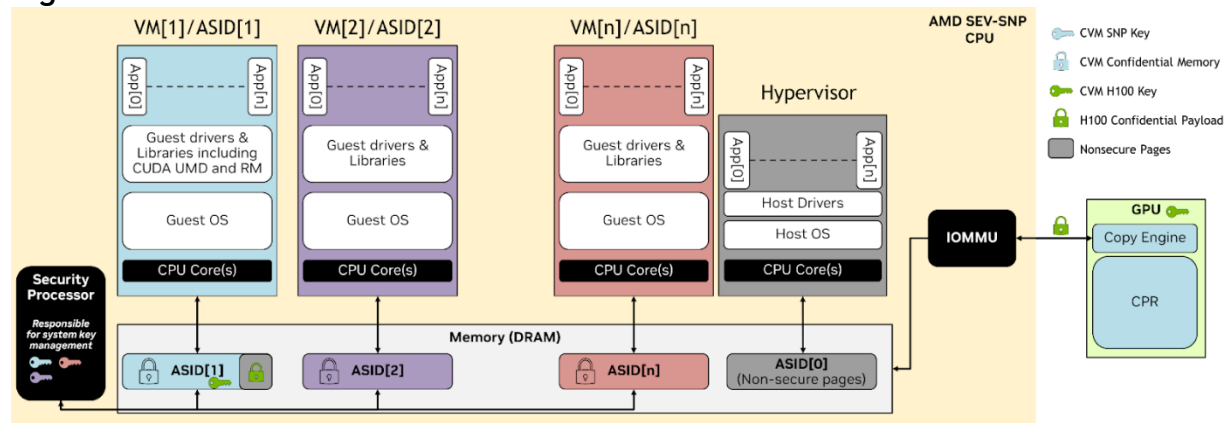
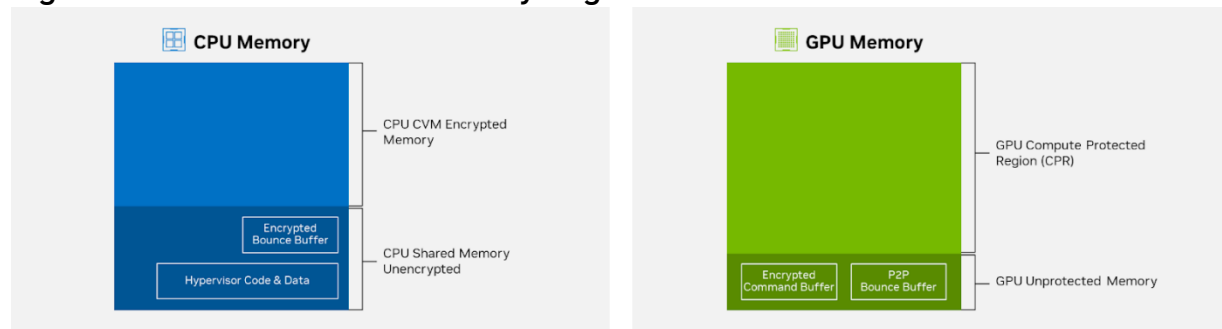


Figure 11 shows an AMD CPU with SEV-SNP, running n CVMs, one of which (VM[1]) has an H100 GPU passed through. Since the GPU cannot directly access the CVM ASID[1]

memory shown in light blue, all communication between the CPU and the GPU must go through a bounce buffer allocated in shared memory (gray box shown in ASID[1]).

Figure 12. Protected Memory Regions for CPU and GPU



The H100 GPU has DMA engines with encrypt/decrypt capability, which are responsible for the movement of data to and from the CPU's memory. In a confidential environment, DMA engines are only allowed to access shared memory pages to retrieve and place data. To ensure the confidentiality and integrity of the payloads, models, and data, the data in these pages are encrypted and signed. These shared memory regions are called *bounce buffers*, because they will be used to stage the secured data before the data is transferred into the secured memory enclaves, is decrypted and authenticated, and then processed. NVIDIA has long provided our developers with a solution called Unified Virtual Memory (UVM) that automatically handles page migrations between the GPU memory and the CPU memory based on a memory allocation API called `cudaMallocManaged()`. When the CPU accesses the data, UVM migrates the pages to the CPU system memory. When the data is needed on the GPU, UVM migrates it to the GPU memory. For CC, we extended UVM to employ encrypted and authenticated paging through bounce buffers in shared memory. For use cases where multiple GPUs might be moving data to and from each other using NVLINK, we employ hardware encrypted transfers through bounce buffers in GPU unprotected memory.

Developer Considerations

Here we summarize some of the considerations that developers must be aware of when using the H100 in CC mode.

- > Due to how CPU vendors isolate their CVM memory from outside sources, pinned memory allocations such as from `cudaHostAlloc()` and `cudaMallocHost()` cannot be directly accessed by the GPU.
- > Instead, they are handled by UVM with encrypted paging, as if they were allocated by `cudaManagedAlloc()`. This means pinned memory accesses are slower in CC mode.
- > `cudaHostRegister()` cannot be supported because this API gives direct access to memory created by `malloc()` or `new()`, inside the CVM.

- > This API, among a few others, will return an error code when the GPU is in CC modes. `cudaHostRegister()` does not have widespread use in NVIDIA libraries, and where it is used, we are in the process of modifying the code paths to work seamlessly with the H100 in CC mode.
- > Developers must use the `nvidia-persistenced` daemon when using the H100 GPU in CC mode to keep the driver loaded, even when not in use.
 - In a typical operation, when the NVIDIA device resources are no longer being used, the NVIDIA kernel driver will tear down the device state. However, in the CC mode, this leads to destroying the shared-secret and the shared keys that were established during the setup SPDM phase of the driver. To protect user data, the GPU will not allow the restart of an SPDM session establishment without an FLR, which resets and scrubs the GPU.
 - `nvidia-persistenced` provides a configuration option called persistence mode that can be set by NVIDIA management software, such as `nvidia-smi`. When the persistence mode is enabled, the NVIDIA kernel driver is prevented from exiting. `nvidia-persistenced` does not use any device resources. It simply sleeps while maintaining a reference to the NVIDIA device state.

Performance Samples

A primary goal of delivering CC to customers is that CUDA applications can run unchanged while maximizing the acceleration potential of the underlying hardware and software. CUDA provides lift and shift benefits to applications that will be run in CC mode. As a result, the NVIDIA GPU CC architecture is compatible with the CPU architectures that also provide application portability from non-confidential to CC environments.

Given the description so far, it should not be surprising that CC workloads on the GPU perform close to non-CC mode when the amount of compute is large compared to the amount of input data. When the amount of compute is low compared to the input data, the overhead of communicating across the non-secure interconnect limits the application throughput. Here is some overview information to help understand performance in CC mode:

- > In CC mode, the following performance primitives are at par with non-confidential mode:
 - GPU raw compute performance

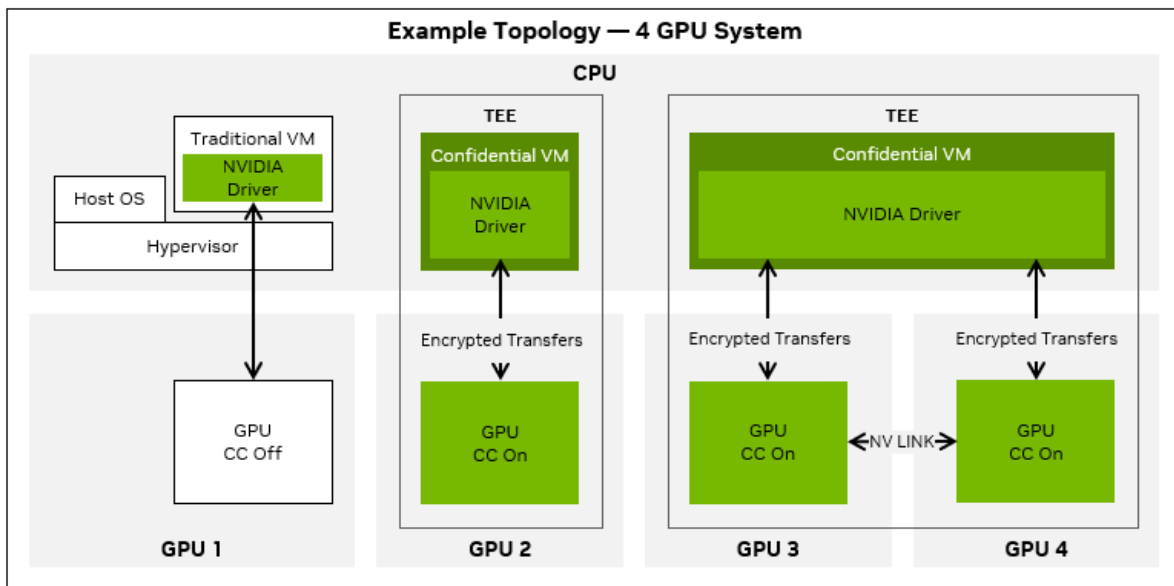
The compute engines execute unencrypted code on unencrypted data resident in GPU memory.
 - GPU memory bandwidth

The on-package HBM memory is considered secure against common physical attack tools, such as interposers, and is not encrypted.
- > The following performance primitives are impacted by additional encryption and decryption overheads:

- CPU-GPU interconnect bandwidth is limited by CPU encryption performance to approximately 4 GBytes/sec.
- Data transfer throughput across the non-secure interconnects incurs a latency overhead for the transit through encrypted bounce buffers in unprotected memory.
- GPU peer memory bandwidth in multi-GPU use cases

Direct connect multi-GPU NVLINK topologies use encrypted bounce buffers in unprotected GPU memory that reduces throughput.

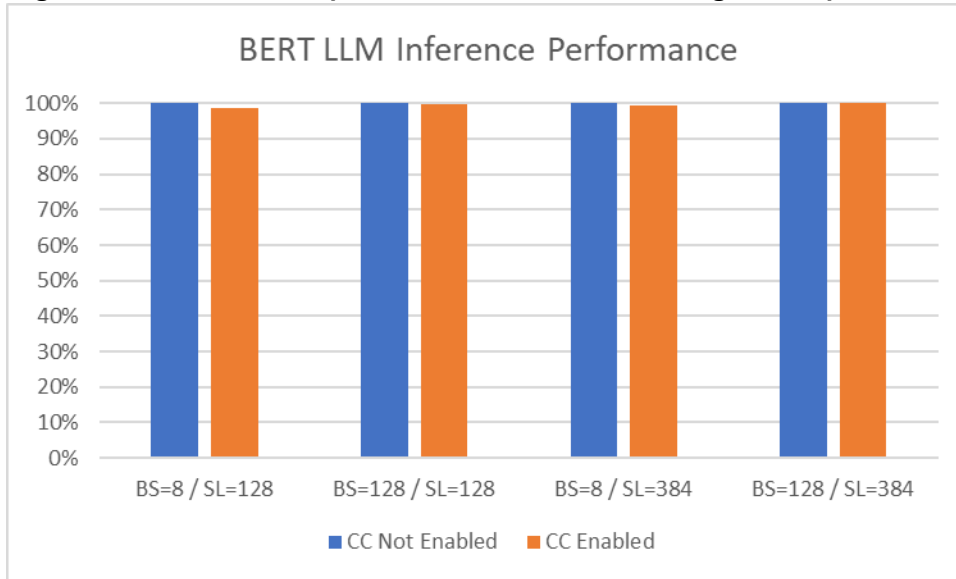
Figure 13. Example Server Topology with GPUs in CC On and Off



There is also an overhead for encrypting GPU command buffers, synchronization primitives, exception metadata, and other internal driver data that is exchanged between the GPU and the confidential VM running on the CPU. Encrypting and authenticating these data structures prevents side channel attacks on the user data.

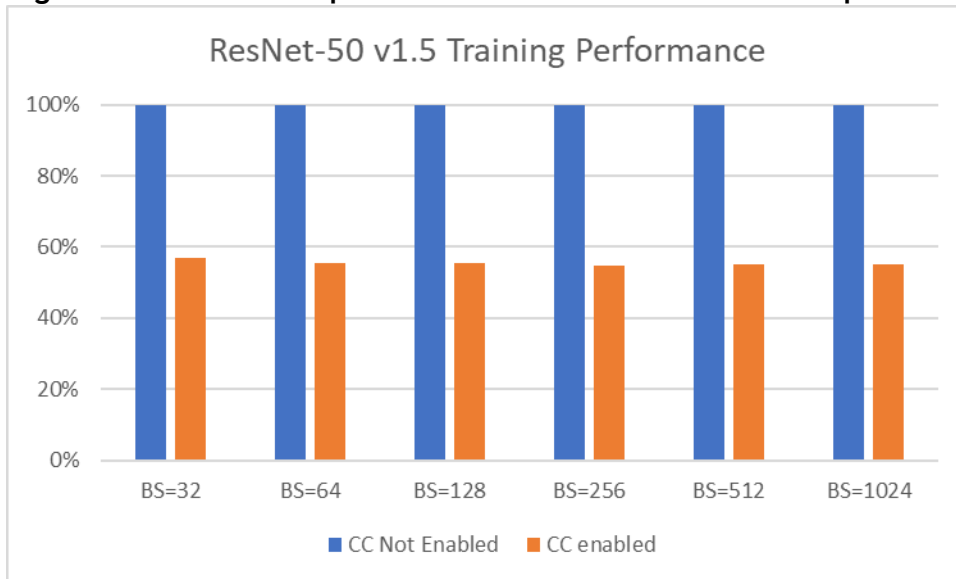
CUDA Unified Virtual Memory has long allowed developers to use the same virtual address pointer from the CPU and the GPU, which really simplifies the application code. In CC mode, the unified memory manager has to encrypt all pages that are being migrated across the non-secure interconnect.

Figure 14. Example of a Workload with High Compute to I/O Ratio



BS is the batch size, and SL is the sequence length.

Figure 15. Example of a Workload with a Low Compute to I/O Ratio



BS is the batch size.

Summary

CC is a paradigm which is rapidly approaching, because of regulatory regimes, privacy concerns, or the desire to accelerate other sensitive workloads. The entire computing industry recognizes the need to modify the traditional thinking and operation of security when operating on data. NVIDIA is at the tip of this spear, collaborating with CPU partners, Cloud Providers, and ISVs to ensure that the change from traditional accelerated workloads to confidential accelerated workloads will inevitably be as transparent as the change from <http://> to <https://>. The need to accelerate high-performance workloads is only going to continue to grow in-line with the equivalent need to ensure those workloads remain secure.

The Hopper H100, as the first Confidential Computing GPU, is primed and ready to accept these workloads. For more information, please feel free to reach out to your NVIDIA contact, or at our [Forum](#).

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.

<ALL Automotive and Embedded documents get this: <Unless specifically agreed to in writing by NVIDIA,> 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 NON-INFRINGEMENT, 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, **<Add all TM/R words included in document>** 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.

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.

Arm

<ALL TEGRA DOCS GET THIS> Arm, AMBA, and ARM Powered are registered trademarks of Arm Limited. Cortex, MPCore, and Mali are trademarks of Arm Limited. All other brands or product names are the property of their respective holders. "Arm" is used to represent ARM Holdings plc; its operating company Arm Limited; and the regional subsidiaries Arm Inc.; Arm KK; Arm Korea Limited.; Arm Taiwan Limited; Arm France SAS; Arm Consulting (Shanghai) Co. Ltd.; Arm Germany GmbH; Arm Embedded Technologies Pvt. Ltd.; Arm Norway, AS, and Arm Sweden AB.

OpenCL

OpenCL is a trademark of Apple Inc. used under license to the Khronos Group Inc.

Copyright

© 2023 NVIDIA Corporation. All rights reserved.