Skalierbare KI/ML-Infrastrukturen

Evaluieren, Automatisieren, Praxis

Kapitel 4
NVIDIA-Datacenter-GPUs und mehr – technischer Background

»9.7 TFLOPS for FP64? Sounds great, but, eh – can it run DOOM?«

– Typischer Foren-Kommentar bei so ziemlich jeder Ankündigung einer neuen High-End-GPU-Karte

In diesem Abschnitt befassen wir uns konkreter mit der Hardware, sprich – den NVIDIA-Datacenter-tauglichen GPUs, den jeweils dafür geeigneten Bereitstellungsverfahren, ihrer Performance und etlichem anderen mehr. Mit dem Bereitstellungsverfahren ist im einfachsten Fall beispielsweise das dedizierte Durchreichen einer GPU an eine VM gemeint oder softwarebasierte GPU-Partitionierungen per Virtual GPU (vGPU), hardwarebasierte Partitionierung (MIG) mit einer besseren Tenancy, GPU-Sharing oder Kopplungen multipler GPUs, lokal oder über das Netzwerk, oder auch verschiedene Kombinationen der vorgenannten Verfahren, um unterschiedlichste Anwendungsfälle abdecken zu können.

4.1 NVIDIA und ML-Cluster

Nun, wo fängt man an? Vielleicht mit einer Frage:

Lieber NVIDIA-Aktien anstelle von Goldbarren? Nein, das soll durchaus kein polemischer Kommentar zu Anlagestrategien sein, sondern liegt näher an der Realität, als Sie vielleicht denken mögen, wie im Folgenden erklärt werden wird. Denn kaum ein Unternehmen hat sich strategisch in den letzten Jahren in eine ähnliche monopolartige Position gebracht, wenn es um KI/ML-Belange im RZ-Bereich geht, wie NVIDIA.

NVIDIAs GPU-Accelerators wie die neue Hopper-Generation oder ihre Vorläufer wie die Ampere-Generation oder deren Tesla-Vorgänger (Turing-Generation) sind aus Rechenzentren kaum noch wegzudenken. Datacenter-taugliche GPUs werden – je nach GPU-Modell – zwar auch noch in bestimmten Unternehmen für Remote-Desktop-Szenarien, also »echte« grafische Workloads eingesetzt, in zunehmender Anzahl jedoch primär nur noch für *Accelerated Computing*, also KI/ML-Einsätze mit GPU-Beschleunigung.

Dabei existieren verschiedene Möglichkeiten, die Rechenleistung der GPU zu nutzen. Während für reine ML-Entwicklungs-Workstations beispielsweise dedizierte High-End-Grafik-

karten wie eine High-End-Karte der RTX-Serie per *Passthrough* angebunden werden können, bleibt das Manko, dass die jeweilige GPU nur von *einem* Nutzer bzw. exakt einer realen oder virtuellen Maschine für ML-Workloads verwendet werden kann. Für geclusterte und skalierbare Szenarien in Unternehmen wird also eine andere Herangehensweise benötigt.

Insbesondere Kubernetes-/OpenShift-basierte Container-Cluster mit Pod- und Cluster-Autoscalern sind daher eigentlich prädestiniert, um bei entsprechender Lastanforderung automatisch in die Breite zu skalieren, um so für alle (KI/ML-)Anwendungsfälle genügend Rechenkapazitäten bereitstellen zu können. So können z. B. multiple GPU-fähige Anwendungen automatisch horizontal skaliert werden, um große Datenlasten parallel prozessieren zu können – natürlich vorausgesetzt, dass die Anwendung bzw. Architektur dies unterstützt.

Logischerweise gibt es auch alternative Verfahren wie das bereits erwähnte, dedizierte Passthrough (1:1-Bindung zwischen VM und GPU-Accelerator), die zwar technisch ebenfalls greifen, jedoch in Clustern (siehe das Thema »Skalierung«) für Accelerated Computing selbst mit Verfahren wie VMwares *Dynamic DirectPath IO* (siehe Abschnitt 4.5.3) nicht wirklich effizient sind. Es werden daher *shareable* GPU-Ressourcen benötigt, und hier startet die Geldruckmaschine für Cloud-Betreiber, Virtualisierer – und nicht zuletzt NVIDIA.

Betrachten wir das »Warum«.

Cloud

Nahezu alle Cloud-Provider lassen sich mittlerweile ihre VM-Instanztypen mit durchgeschleifter GPU/TPU-Accelerator-Hardware hervorragend vergüten, wie in Abschnitt 3.2 bereits betrachtet. Sicher, die üblicherweise horrenden Preise für RZ-taugliche GPU-Accelerators stellen auch für Cloud-Provider zunächst eine Investition dar, die sich jedoch auf deren Seite sehr schnell amortisiert. (v)GPU-Manager, die NVIDIA-GPUs softwaretechnisch auf (Public-)Cloud-Hypervisoren wie GCP oder AWS partitionieren könnten, waren im betrachteten Stand für Endverbraucher nicht verfügbar.

Virtualisierer

Nehmen wir VMware als Beispiel und die in den meisten Unternehmens-RZ präsenten vSphere-Systeme: Schon interessant, dass gerade das *vGPU*-Feature, das anfänglich kaum Beachtung fand, nun nur noch in VMwares höchster Lizenz-Preisklasse zu finden ist. Ohne vSphere Enterprise Plus-Lizenz läuft im Bereich »*vGPU-Computing*« (vorab: softwarebasierte Partitionierung einer GPU) rein gar nichts. Und betrachten wir den »Aufwand«, der dahintersteckt, ist die Verortung dieses Features in der höchsten Lizenz-Preisklasse ohne jedweden technischen Hintergrund, sondern rein gewinnorientiert motiviert. Denn es handelt sich auf ESXi-Seite lediglich um die Integration eines einzigen VIBs, das zudem noch von NVIDIA geliefert wird. Dagegen ist prinzipiell nichts einzuwenden, das ist freie Marktwirtschaft. Aber dennoch wäre hier seitens VMware eine etwas niedrigere finanzielle Einstiegshürde gerade für KMU durchaus wünschenswert gewesen. Aber wozu den Profit minimieren, wenn es die GPU-Quelle #1 nicht anders vormacht.

NVIDIA

Sicher, leistungsstarke GPU-Karten waren noch nie günstig, und einige Hardcore-Gamer würden wahrscheinlich lieber eine Hypothek aufnehmen und/oder Organe spenden, als das neue Spiel XYZ nicht in Ultra-Super-HD-256K-with-unbelievable-high-FPS zu spielen. Aber erst seit ihrer neu erstrahlenden Popularität im KI/ML-Bereich schießen die Preise für »echte« GPU-Accelerators wie NVIDIAs Ampere- oder Tesla-Modelle (oder RTX-Karten für ML-Workstations) durch die Decke – fünfstellige Beträge für RZ-taugliche Karten der Oberliga sind hier keine Seltenheit, sondern das Normalprogramm, und komplette DGX-Systeme liegen wie bereits betrachtet noch deutlich darüber. Wer denkt, dass sein Unternehmen mit gemieteten, cloudgehosteten, RZ-tauglichen GPUs auf Dauer günstiger fährt – eine nette Idee, die oft nur dann aufgeht, wenn es sich nicht um 24/7-Dauerbetrieb der Systeme handelt.

4.2 Partitionierte GPUs mit NVIDIAs vGPU und MIG

Betrachten wir an dieser Stelle zunächst einige grundlegende Begrifflichkeiten rund um die möglichen Bereitstellungsarten RZ-tauglicher GPU-Hardware, da sich diese zum Teil grundlegend von »normalen« Verfahren im HEDT/Workstation-Bereich unterscheiden und wichtig für das spätere Verständnis der durchzuführenden Setup-Tasks sind.

Ein wichtiger Aspekt ist dabei die Möglichkeit, GPUs zu *partitionieren* und damit ihre immense Rechenleistung, die sonst gegebenenfalls nicht vollumfänglich genutzt wird, an multiple Client-Prozesse zu verteilen. Dazu stehen verschiedene Verfahren zur Verfügung, die wir im Folgenden betrachten.

Eine weitere Variante, die seit dem GPU-Operator 1.11 zur Verfügung steht, um Däumchen drehende GPUs in KI/ML-Clustern zu vermeiden, ist die neue Option des GPU-Sharings/Overcommitments (z. B. physikalisch 1 GPU, aber *n* Client-Prozesse dürfen die GPU beanspruchen). Dies wird konkret ab Abschnitt 10.10 betrachtet.

4.2.1 Vorbetrachtungen und Scope

Die primär für grafische Anwendungsfälle gedachten GPU-Modi *vSGA* und *vDGA* werden im Folgenden nicht angesprochen, da sie für den Scope des Buches irrelevant sind.

Für skalierbaren Datacenter-Betrieb mit Fokus auf Compute-Anwendungen (KI/ML) und multiple Tenants sind üblicherweise nur die Modi *vGPU* (vGPU = virtual GPU, »Software«-partitionierbare GPUS) und *MIG* (MIG = Multi Instance GPUs = »Hardware«-partitionierbare GPUs, die üblicherweise auch den vGPU Modus unterstützen) geeignet und von Interesse, in wenigen Ausnahmenfällen auch ein PCI-Passthrough von ganzen GPUs ohne Partitionen.

Daher liegt der Fokus aller folgenden Betrachtungen auf den GPU-Bereitstellungsmodi vGPU und MIG und den damit verbundenen Verfahren und Konzepten für einen effizienten Datacenter Betrieb.

4.2.2 Vorbetrachtungen: Partitionierte GPUs mit vGPU und MIG

Die Partitionierung einer physikalischen GPU ist besonders effizient für Workloads, die die Rechenkapazität der physikalischen GPU nicht vollständig auslasten würden. Je nach Leistungsklasse der GPU und dem konsumierenden Workload wird in der Praxis daher oftmals keine vollständige GPU benötigt. Dies gilt insbesondere dann, wenn es sich um kleinere PoC/Modelle oder Demos handelt, die auf einem Boliden wie einer A100 mit 40 oder gar 80 GB GPU-RAM betrieben werden sollen.

Ohne die Möglichkeit einer GPU-Partitionierung wird jeder ML-Workload die komplette GPU okkupieren, unabhängig davon, ob er sie tatsächlich verwendet bzw. voll auslastet … oder eben nicht. Insbesondere containerisierte Compute-Workloads im RZ/Cloud-Umfeld auf Kubernetes-/OpenShift-Clustern profitieren von der höheren Effizienz einer GPU-Partitionierung sowie der im Fall von MIG damit einhergehenden, echten Mandanten-Fähigkeit bzw. Multi-Tenancy und damit höheren Sicherheit.

NVIDIAs vGPU-(Virtual-GPU-)Software ermöglicht bereits seit langem die softwarebasierte/temporäre Partitionierung von geeigneten NVIDIA-GPUs und unterstützt auch eine breite Palette an weit verbreiteten und relativ kostengünstigen Datacenter-GPUs, wie z. B. Tesla T4-Instanzen. Für den vGPU-Modus wird auf Hypervisor-Ebene ein lizenzpflichtiger GPU-Manager benötigt.

Der *MIG-Mode (Multi-Instance GPU)*, der mit NVIDIAs Ampere-GPU-Generation und Modellen wie der A100 oder A30 erstmals zur Verfügung stand, ist zwar ebenfalls ein Betriebsmodus um GPUs zu partitionieren, aber im Gegensatz zum vGPU Mode bringt dieses Verfahren eine deutlich effizientere Multi-Tenancy, die jedoch im Vergleich zum vGPU-Mode meist mit einer höheren Idle Power einhergeht, siehe Abschnitt 5.4.4. MIG-fähige Karten können in der Regel auch eine Partitionierung mittels vGPU anbieten. Die (vorab vereinfacht und abstrakt ausgedrückt) GPU-Core-Tenancy ist dann jedoch nicht mehr aktiv. Alle Details zum MIG-Modus finden Sie ab Abschnitt 4.4.

Während sich im *vGPU-Modus* (Time-Sliced, CPU-like Scheduling) alle GPU-Partitionen im gleichen Hardware-Pool der GPU-Cores-Engine bedienen und nur den GPU-RAM partitionieren, sorgt der *MIG-Mode* für explizite Separierung der multiplen (im betrachteten Stand bis zu sieben) GPU-Instanzen. Dadurch können Workloads besser aufgeteilt und zudem die Datensicherheit erhöht werden. Sie hierzu auch: *https://www.nvidia.com/de-de/technologies/multi-instance-gpu/*

MIG-fähige Karten werden von NVIDIAs GPU-Operator ab Version 1.7 unterstützt.

4.2.3 NVIDIAs vGPU und (leider noch kein) Cloud-Einsatz

Für alle folgenden Betrachtungen ist es vorab essentiell wichtig zu verstehen, dass *NVIDIAs vGPU-Software* (der GPU-Manager) auf gängigen Public-Cloud-Plattformen wie GCP, AWS oder Azure typischerweise *nicht* zur Verfügung steht. Der GPU-Manager (und damit vGPU) steht nur für Hypervisoren zur Verfügung, wie sie in Unternehmens-RZ bzw. Private Clouds eingesetzt werden, wie z. B. vSphere, Citrix, RHEL KVM, Nutanix und andere.

Hintergrund: Der *GPU-Manager* wird bereits auf der Ebene des Hypervisors zwingend für vGPU benötigt, damit über ihn GPUs softwaretechnisch partitioniert und an die VMs durchgereicht werden können. So viel Zugriff kann oder will im betrachteten Stand keiner der Public-Cloud-Riesen dem Endkunden anbieten. In logischer Konsequenz existiert daher (bisher) auch keine GPU-Manager-Software seitens NVIDIA für AWS, GCP und Co. Somit bleiben den Unternehmen, die GPU-Workloads in der Cloud prozessieren möchten oder müssen, nur zwei Alternativen.

Entweder werden nicht-MIG-fähige Karten, wie z. B. T4, A10 oder A40 komplett (en bloc, d. h. 1:1) via Passthrough direkt an den Consumer (die KI/ML-Applikation) durchgereicht oder es kommen – wenn Tenancy gefragt ist und/oder multiple Consumer gleichzeitig auf einer großen, leistungsfähigen GPU wie der A100 sicher bedient werden sollen – MIG-fähige Karten zum Einsatz. Diese werden vom Cloud-Provider typischerweise per Passthrough durchgereicht, können aber vom Kunden bzw. dem GPU-Operator-Stack zumindest auf dem betreffenden Node in GPU-Slices partitioniert und damit von *n* Applikationen verwendet werden.

Auch wenn die Passthrough/MIG-Only-Diktatur der Cloud-Provider die Konfigurationsflexibilität deutlich einschränkt – einen kleinen Vorteil für den Endkunden bietet die vGPU-Abstinenz im Cloud-Umfeld dennoch: Die vGPU-Lizenzkosten entfallen.

4.3 vGPU – Virtual GPU

Schauen wir nun etwas tiefer in Konzepte und Arbeitsweise der vGPU-Partitionierung. Im Folgenden wird zunächst betrachtet, was genau unter vGPU zu verstehen ist, welche Produkte NVIDIA in diesem Bereich anbietet und wie vGPU konzeptionell und technisch betrachtet arbeitet.

4.3.1 Generelle vGPU-Architektur

Betrachten wir zunächst die generelle vGPU-Architektur anhand Abbildung 4.1, die eine generelle, vereinfachte Darstellung der Arbeitsweise einer vGPU und ihre Aufteilung an multiple VM-Instanzen zeigt.

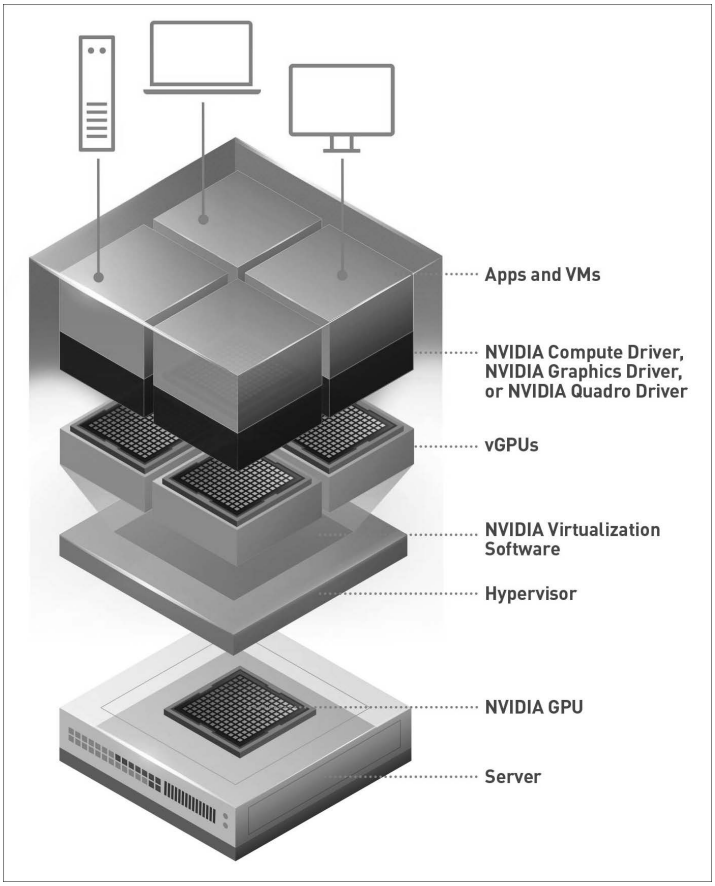

Abbildung 4.1 NVIDIAs vGPU-Stack in stark vereinfachter, konzeptioneller Form, Quelle: NVIDIA

4.3.2 Details zur Funktionsweise

Wie in Abbildung 4.1 bereits zu erkennen ist, wird die vGPU-Funktionalität als zusätzliche Software-Komponente zwingend über den jeweils verwendeten *Hypervisor* bereitgestellt. Im betrachteten Stand wurden u. a. folgende (in der Regel Typ1-)Hypervisoren/Bare-Metal-OS unterstützt (Nennung hier ohne Versionen): vSphere, OpenStack, Windows Server/Hyper-V, RHEL KVM/RHV, XenServer.

Es ist ebenfalls zu beachten, dass Ampere-Karten wie z. B. die A30 oder ihr größeres Pendant, die A100 oder die neue H100, die per *MIG* (siehe ab Abschnitt 4.4) auch hardwarebasiert (korrekter wäre »räumlich«/»EEPROM-like«) partitioniert werden können, ebenfalls rein softwarebasiert im *vGPU*-Mode betrieben werden können. Wann welcher Modus (vGPU, MIG oder MIG-backed vGPU) zum Einsatz kommt, bestimmen üblicherweise die (gegebenenfalls vorgegebene) RZ-Architektur, Security-Anforderungen und der konkrete, technische Einsatzzweck.

Bleiben wir zunächst bei der *vGPU-Partitionierung*. Jede GPU-Karte stellt eine bestimmte Anzahl an Rechenkernen und verfügbarem Grafikspeicher (RAM) zur Verfügung. Diese Compute-Ressourcen können nun an eine VM (oder eben partitioniert an mehrere) durchgeschleift werden. Während es sich bei den – von der jeweiligen VM abgerufenen – (v)GPU-Compute-Ressourcen wie üblich um eine komprimierbare Compute-Ressource handelt (vergleiche CPU/GPU-Shares), stellen die RAM-Ressourcen wie üblich ein nicht komprimierbares Hard-Limit dar. Ein einfaches Rechenbeispiel:

Reichen wir eine vGPU-fähige Karte mit 16 GB RAM an vier VMs bei identischer Partitionierung durch, so erhält jede VM exakt 4 GB GPU-RAM. Damit sind die nicht komprimierbaren bzw. nicht over-commit-fähigen Compute-Ressourcen dieser GPU-Karte erschöpft.

Achtung: vGPU und GPU-Pool-Sharing

Dabei ist jedoch zu beachten, dass die partitionierten GPU-Slices auf einer nicht-MIG-, sondern »nur« vGPU-fähigen Karte nicht wirklich voneinander isoliert sind. Alle vGPU-Partitionen z. B. einer Tesla T4 oder A10 nutzen die GPU-Ressourcen wie einen »Shared«-Pool. Wer zwingend echte *Tenancy* braucht, wird um MIG-fähige Karten nicht herumkommen.

Das vGPU-Modell bietet dabei die funktionalen Eigenschaften, die Sie in Abbildung 4.2 sehen.

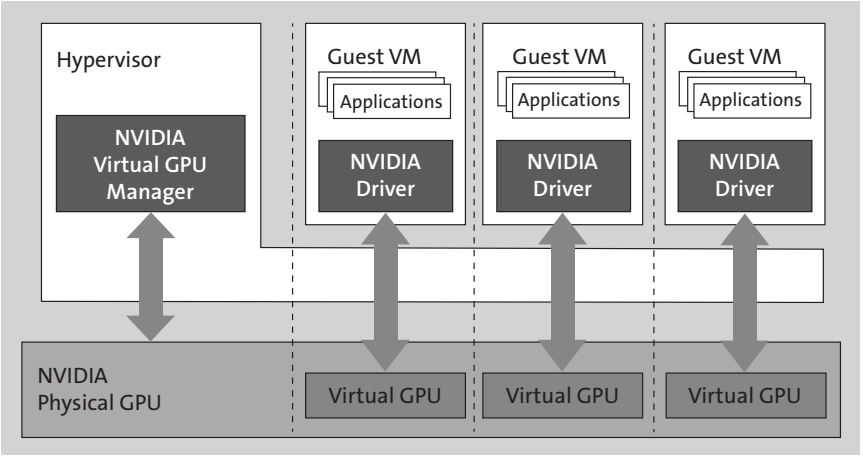

Abbildung 4.2 NVIDIA-vGPU-Systemarchitektur – High Level, Quelle: NVIDIA

Jede NVIDIA-*vGPU* kann analog zu einer herkömmlichen GPU betrachtet werden: Sie besitzt eine fixe/statische Menge an GPU-Framebuffer (RAM der GPU-Karte) und einen oder mehrere virtuelle Displayausgänge oder auch Heads.

Die virtuelle GPU (*vGPU*) erhält zum Zeitpunkt ihrer Bereitstellung exklusiv einen Teil des Framebuffers der realen GPU (oder je nach Profil gegebenenfalls auch den ganzen). Diesen verwendet die virtuelle GPU so lange, bis sie deaktiviert wird (z. B. Shutdown der VM).

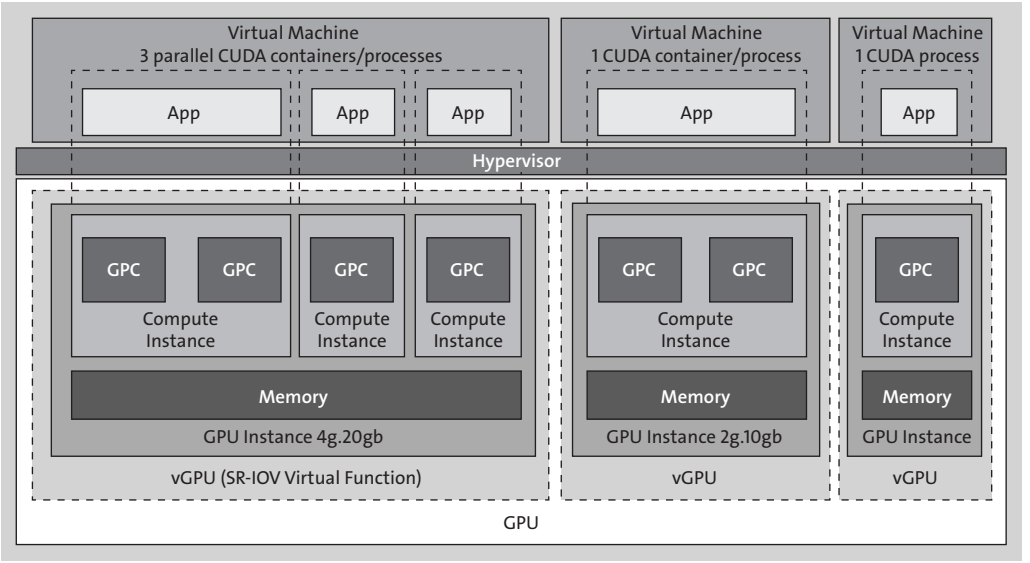

Abbildung 4.3 NVIDIA-vGPU-Systemarchitektur – Details, Quelle: NVIDIA

4.3.3 Time-Sliced vGPU

Der vGPU-Modus (ohne explizite Abschottung im GPU-Engine-Layer) ist mit jeder NVIDIA-vGPU-fähigen Karte verfügbar (siehe Abbildung 4.4).

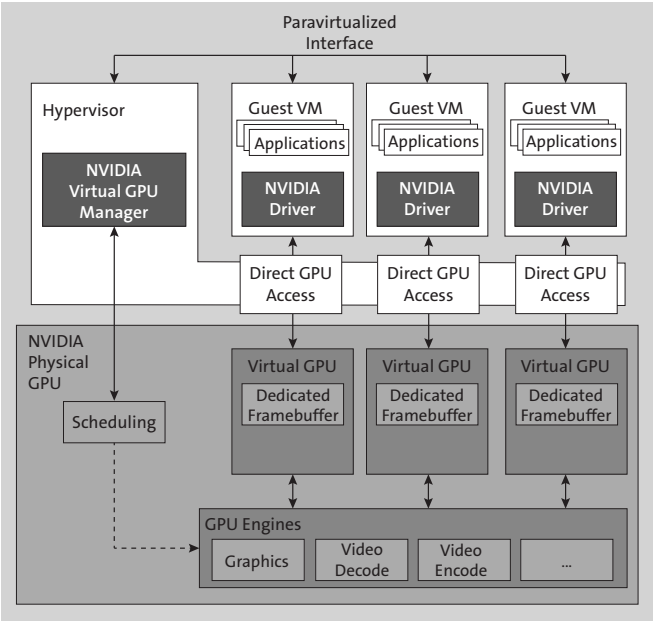

Abbildung 4.4 NVIDIA-vGPU-Systemarchitektur – Time-Sliced, Quelle: NVIDIA

vGPU-fähige Karten

Eine Auflistung vGPU-fähiger Karten von NVIDIA findet sich unter anderem in der Dokumentation NVIDIAs, wird aber auch noch in anderen Abschnitten des Buches thematisiert (u. a. Abschnitt 5.3.1): *https://docs.nvidia.com/grid/gpus-supported-by-vgpu.html*.

4.3.4 Passthrough GPU vs. vGPU im Hypervisor

Abbildung 4.5 zeigt im direkten Vergleich, wie GPUs direkt per Passthrough oder per vGPUs auf einem vSphere-System an die darüberliegenden virtuellen Clients durchgereicht werden.

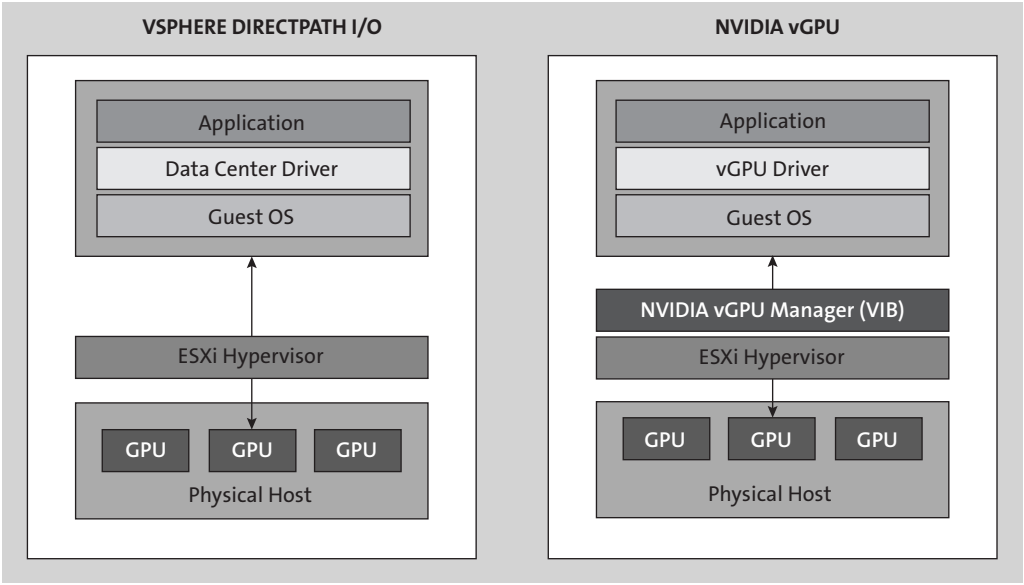

Abbildung 4.5 Passthrough GPU vs. vGPU-Workflow zwischen VM und ESXi-Hypervisor, Quelle: VMware

Im Passthrough-Mode (links) wird das PCI-Device über den Hypervisor dediziert an einen Consumer (sprich: VM) durchgereicht. Soll stattdessen der vGPU-Mode (rechts) zum Einsatz kommen, wird auf dem Hypervisor ein GPU-Manager in Form eines VIB installiert. Dieses registriert die verbauten NVIDIA-GPUs und stellt bei der Einrichtung einer VM oder eines Templates die Möglichkeit zur Verfügung, nur bestimmte Partitionen einer GPU an eine VM durchzureichen. In der VM muss bei diesem Setup-Typ auch ein passender, vGPU-fähiger Treiber installiert werden.

Diese Installation der GPU-Treiber (egal ob normaler Driver oder vGPU) erfolgt in containerisierten Umgebungen üblicherweise vollautomatisch über den GPU-Operator, der ab Abschnitt 9.4 besprochen wird.

4.3.5 vGPU-Produkte

Die Rubrik *vGPU* umfasste gemäß der Definition von NVIDIA im Stand 07/2022 die folgenden Produkte:

- *NVIDIA Virtual Applications* (*vApps*) – Dieses vGPU-Produkt ist z. B. für Unternehmen gedacht, die Citrix Virtual Apps und Desktops bereitstellen oder RDSH oder andere App-Streaming- oder sessionbasierte Lösungen einsetzen. Generell: für Anwendungen auf PC-Ebene und serverbasierte Desktops geeignet.
- *NVIDIA Virtual PC* (*vPC*) – Für Anwendungsfälle, in denen virtuelle Remote-Desktops (»virtuelle PCs«) mit PC-Windows-Anwendungen, Browsern und High-Definition-Video benötigt werden.
- *NVIDIA RTX Virtual Workstation* (*vWS*) – Für den Einsatzfall, remote grafische Systeme (z. B. CAD/CAE) mit maximaler Leistung zu betreiben.
- *NVIDIA Virtual Compute Server* (*vCS*) – Der von uns primär betrachtete Anwendungsfall: HPC-/KI-/ML-/Deep-Learning-Workloads ohne grafische Ausgabe (*Display-less*), die z. B. in VMs/Containern auf Hypervisoren (vSphere, KVM etc.) mit ECC-Unterstützung betrieben werden können.

Siehe dazu auch: *https://www.nvidia.com/content/dam/en-zz/Solutions/design-visualization/solutions/resources/documents1/Virtual-GPU-Packaging-and-Licensing-Guide.pdf*.

4.3.6 vGPU-Versionen

Es ist essenziell wichtig zu verstehen, dass *NVIDIAs vGPU-Software-Version* (im betrachteten Stand 08/2022: vGPU-Version 14.2) immer der Dreh- und Angelpunkt für alle – von dem jeweiligen Release – unterstützten Features ist. Sie allein entscheidet, welche Version des *vGPU-Managers* (Hypervisor) und des *Linux-* und *Windows-Drivers* unterstützt wird und welche jeweiligen Features auf welcher GPU-Hardware und Hypervisor-Plattform supportet werden:

- *https://docs.nvidia.com/grid/index.html*
- *https://docs.nvidia.com/grid/latest/product-support-matrix/index.html*

LTS und (Short-Term) Production Branch

Wie in allen produktivtauglichen Konzepten bietet NVIDIA neben *Short-Term Releases* (unverständlicherweise von NVIDIA als *Production Branch* gekennzeichnet) auch *LTS Releases* (*Long Term Support Branch*). Leider zieht NVIDIA hier keinen »roten Faden« bezogen auf die Nummerierung (etwa durch gerade und ungerade Nummern) durch.

Einige der aktuelleren, ungeraden Release-Nummern werden für die LTS-Versionen verwendet (11, 13), einige der geraden (12, 14) für die Short-Term Branches. In älteren Versionen passt dieses Schema nicht, daher kontrollieren Sie immer den verwendeten Branch gemäß der Support-Matrix.

Ein vGPU *LTS Branch* hatte im betrachteten Stand einen Supportzeitraum von drei Jahren: So ist der EOL für den vGPU 13.x Branch, der im August 2021 gestartet wurde, im August 2024 erreicht. Die *Short-Term Branches* hingegen haben in der Regel nur einen maximalen Supportzeitraum von einem Jahr.

Hinweis: NVIDIA AI Enterprise (NVAIE-Versionen)

Bitte beachten Sie bereits an dieser Stelle, dass sich vGPU- und NVAIE-Software (Letztere ab Kapitel 6 erläutert) in Art, Aufbau und Lizenzierung (siehe ab Kapitel 7) unterscheiden. Die von NVAIE unterstützten Produkte und Features finden sich hier:

https://docs.nvidia.com/ai-enterprise/latest/release-notes/index.html.

4.3.7 vGPU-Features je nach Produkt

Tabelle 4.1 listet die vGPU-Features je nach gewähltem/lizenziertem Produkt auf. Bezogen auf den Scope des Buches ist für uns primär nur die letzte Spalte der Tabelle (vCS, virtual Compute Server) von Relevanz.

Feature	NVIDIA vApps	NVIDIA vPC	NVIDIA vWS	NVIDIA vCS
License Entitlement				
Concurrent User (CCU)	✓	✓	✓	
Per GPU				✓
Capability Entitlement				
Desktop Virtualization		✓	✓	
RDSH App Hosting	✓	✓	✓	
RDSH Desktop Hosting	✓	✓	✓	
Compute Virtualization			✓	✓
Windows Guest OS Support	✓	✓	✓	

Tabelle 4.1 vGPU-Features und -Lizenzen

Feature	NVIDIA vApps	NVIDIA vPC	NVIDIA vWS	NVIDIA vCS
Linux Guest OS Support		✓	✓	✓
Maximum Displays	1	4	4	1
Maximum Resolution	1280 × 1024	5120 × 2880 (5K)	7680 × 4320 (8K)	4.096 × 2160 (4K)
NVIDIA RTX Enterprise Software Features			✓	
OpenGL, DirectX and Vulkan	✓	✓	✓	In-situ Graphics only
CUDA and OpenCL Support			✓	✓
ECC and Page Retirement			✓	✓
Multi-vGPU			✓	✓
NVLink			✓	✓
GPU Passthrough Support	✓		✓	✓
Bare Metal Support	✓		✓	
vGPU Profile Sizes Supported				
512 MB		✓	✓	
1 GB	✓	✓	✓	
2 GB	✓	✓	✓	
3 GB	✓		✓	
4 GB	✓		✓	✓
5 GB				✓
6 GB	✓		✓	✓

Tabelle 4.1 vGPU-Features und -Lizenzen (Forts.)