AYI NEDJIMI Consultants

Proxmox GPU Passthrough — NVIDIA et AMD pour IA/ML

Catégorie : Virtualisation Lecture : 16 min Publié le : 12/04/2026 Auteur : Ayi NEDJIMI

Passthrough GPU Proxmox VE pour IA/ML : IOMMU, vfio-pci, NVIDIA CUDA, AMD ROCm, SR-IOV vGPU, LXC GPU, Ollama/vLLM. Benchmarks et troubleshooting inclus.

Le GPU passthrough sur Proxmox VE permet d'attribuer un GPU physique directement à une machine virtuelle, contournant l'hyperviseur pour offrir des performances quasi natives à la VM. Cette technique est devenue incontournable avec l'explosion des workloads d'intelligence artificielle et de machine learning qui nécessitent un accès direct aux capacités de calcul parallèle des GPU NVIDIA (CUDA) et AMD (ROCm). La configuration n'est pas triviale : elle implique la manipulation des groupes IOMMU, le binding du driver vfio-pci, la gestion des ROM GPU et la résolution de problèmes spécifiques à chaque famille de cartes. Ce guide couvre la configuration complète du passthrough pour les GPU NVIDIA (séries RTX, A100, H100, L40S) et AMD (Instinct MI250, MI300, Radeon Pro), incluant le SR-IOV pour le partage de GPU entre VMs, l'accès GPU dans les conteneurs LXC pour les déploiements légers, et les benchmarks de performance comparés au bare metal. Les configurations présentées sont testées et validées sur Proxmox VE 8.2 et 9.0.

Prérequis : IOMMU et VT-d/AMD-Vi

Le passthrough PCI repose sur l'IOMMU (Input-Output Memory Management Unit), une fonctionnalité matérielle du processeur qui isole les accès DMA des périphériques PCI. Sans IOMMU, un périphérique passé à une VM pourrait accéder à la mémoire de l'hôte ou d'autres VMs, créant un risque de sécurité majeur et des corruptions de données. Pour les aspects sécurité de la virtualisation, consultez notre guide de migration VMware vers Proxmox.

Activation dans le BIOS/UEFI

Plateforme	Paramètre BIOS	Emplacement typique
Intel	VT-d (Intel Virtualization for Directed I/O)	Advanced > CPU Configuration ou Chipset
AMD	AMD-Vi / IOMMU / SVM Mode	Advanced > NBC Configuration ou IOMMU
Serveur Dell	SR-IOV Global Enable + VT-d	System BIOS > Integrated Devices
Serveur HPE	Intel VT-d + SR-IOV	System Configuration > BIOS Settings

Configuration du kernel Proxmox

```
# Éditer les paramètres GRUB
nano /etc/default/grub

# Pour CPU Intel :
GRUB_CMDLINE_LINUX_DEFAULT="quiet intel_iommu=on iommu=pt"

# Pour CPU AMD :
GRUB_CMDLINE_LINUX_DEFAULT="quiet amd_iommu=on iommu=pt"

# Mettre à jour GRUB
update-grub

# Charger les modules VFIO au boot
cat >> /etc/modules << 'EOF'
vfio
vfio_iommu_type1
vfio_pci
vfio_virqfd
EOF

# Redémarrer
reboot

# Vérifier que l'IOMMU est actif
dmesg | grep -i iommu
# Doit afficher : "DMAR: IOMMU enabled" ou "AMD-Vi: IOMMU enabled"
```

Isolation du GPU : binding vfio-pci

Pour passer un GPU à une VM, il faut d'abord empêcher le pilote graphique de l'hôte (nouveau, nvidia, amdgpu) de s'accaparer le GPU. Le driver vfio-pci prend le contrôle du périphérique au boot et le réserve exclusivement pour le passthrough.

```
# Identifier le GPU et son groupe IOMMU
lspci -nn | grep -i "vga\|3d\|display"
# Exemple : 01:00.0 VGA compatible controller [0300]: NVIDIA Corporation AD102 [GeForce
RTX 4090] [10de:2684]
# Exemple : 01:00.1 Audio device [0403]: NVIDIA Corporation [10de:22ba]

# Vérifier le groupe IOMMU (TOUS les devices du groupe doivent être passés)
find /sys/kernel/iommu_groups/ -type l | sort -t/ -k5 -n | grep "01:00"

# Créer la configuration vfio-pci
echo "options vfio-pci ids=10de:2684,10de:22ba" > /etc/modprobe.d/vfio.conf

# Blacklister les drivers natifs
echo "blacklist nouveau" >> /etc/modprobe.d/blacklist.conf
echo "blacklist nvidia" >> /etc/modprobe.d/blacklist.conf
# Pour AMD : blacklist amdgpu et radeon

# Régénérer l'initramfs
update-initramfs -u -k all

# Redémarrer et vérifier
reboot
lspci -nnk -s 01:00.0
# Kernel driver in use: vfio-pci  ← c'est bon
```

> **Point critique : groupes IOMMU**
> Tous les périphériques appartenant au même groupe IOMMU doivent être passés ensemble à la même VM. Si votre GPU partage un groupe avec le contrôleur USB ou le chipset audio de la carte mère, vous devrez soit passer tous ces périphériques, soit utiliser les **ACS override patches** pour éclater le groupe (avec les implications de sécurité associées). Les cartes mères serveur (Supermicro, ASUS WS) offrent généralement un meilleur isolement IOMMU que les cartes grand public.

Passthrough NVIDIA : CUDA et workloads ML

Configuration de la VM

```
# Créer la VM avec les options nécessaires
qm create 200 \
  --name gpu-ml-worker \
  --memory 32768 \
  --cores 16 \
  --cpu host \
  --bios ovmf \
  --efidisk0 local-zfs:1,efitype=4m \
  --machine q35 \
  --net0 virtio,bridge=vmbr0

# Ajouter le GPU en passthrough
qm set 200 -hostpci0 01:00,pcie=1,x-vga=0

# Pour les GPU NVIDIA avec CUDA (pas besoin de x-vga pour le calcul)
# x-vga=1 uniquement si vous voulez l'affichage graphique dans la VM

# Configuration mémoire pour gros modèles ML
qm set 200 --balloon 0  # Désactiver le ballooning pour les workloads ML
qm set 200 --hugepages 1048576  # Hugepages 1GB pour meilleures performances
```

Installation des drivers NVIDIA dans la VM

```
# Dans la VM Ubuntu/Debian
apt install build-essential linux-headers-$(uname -r) -y

# Installer le driver NVIDIA (méthode recommandée)
apt install nvidia-driver-560 nvidia-cuda-toolkit -y

# OU via le .run officiel (pour contrôle précis de la version)
wget https://developer.download.nvidia.com/compute/cuda/12.6.0/local_installers/
cuda_12.6.0_560.28.03_linux.run
chmod +x cuda_12.6.0_560.28.03_linux.run
./cuda_12.6.0_560.28.03_linux.run

# Vérifier le fonctionnement
nvidia-smi
# Doit afficher le GPU avec la mémoire disponible
```

Contourner le "Code 43" NVIDIA

Historiquement, NVIDIA bloquait le passthrough sur les GPU grand public (GeForce) avec une erreur "Code 43" quand le driver détectait un environnement virtuel. Depuis les drivers 465+ (2021) et la série RTX 30xx+, cette restriction a été officiellement levée. Pour les anciennes cartes, les workarounds suivants restent nécessaires :

```
# Dans /etc/pve/qemu-server/200.conf — uniquement pour GPU pre-RTX 30xx
args: -cpu host,kvm=off,hv_vendor_id=proxmox
# kvm=off masque l'hyperviseur au driver NVIDIA
```

Passthrough AMD : spécificités

Les GPU AMD (Radeon Pro, Instinct) sont généralement plus simples à passer en passthrough car AMD n'impose pas de restrictions artificielles sur la virtualisation. Le driver ROCm pour le calcul ML est open source, ce qui facilite le débogage. Cependant, les GPU AMD souffrent d'un problème connu de "reset bug" : certaines cartes ne se réinitialisent pas correctement lors du redémarrage de la VM, nécessitant un reboot complet de l'hôte.

```
# Passthrough AMD Instinct MI250
qm set 200 -hostpci0 41:00,pcie=1

# Pour contourner le reset bug AMD (si présent)
echo "options vfio-pci enable_sriov=0" >> /etc/modprobe.d/vfio.conf

# Vendor-reset module (solution communautaire pour le reset bug)
apt install pve-headers-$(uname -r) git dkms -y
git clone https://github.com/gnif/vendor-reset.git
cd vendor-reset && dkms install .

echo "vendor-reset" >> /etc/modules
update-initramfs -u
```

SR-IOV et vGPU pour le partage de GPU

Le SR-IOV (Single Root I/O Virtualization) permet de partitionner un GPU physique en plusieurs fonctions virtuelles (VF), chacune assignable à une VM différente. NVIDIA propose cette fonctionnalité via ses GPU datacenter (A100, H100, L40S) avec le profile MIG (Multi-Instance GPU). Pour les déploiements d'IA nécessitant une allocation dynamique des ressources GPU, consultez notre guide d'optimisation des coûts d'inférence LLM.

```
# NVIDIA MIG sur A100 (80GB)
# Activer MIG mode
nvidia-smi -i 0 -mig 1
# Redémarrer le driver
nvidia-smi -r

# Créer des instances GPU (exemple : 3x 20GB)
nvidia-smi mig -i 0 -cgi 9,9,9 -C

# Lister les instances
nvidia-smi mig -i 0 -lgi
nvidia-smi mig -i 0 -lci

# Chaque instance apparaît comme un device séparé utilisable par un conteneur ou une VM
```

Accès GPU dans les conteneurs LXC

Pour les workloads ne nécessitant pas une isolation VM complète, le passthrough GPU dans un conteneur LXC offre des performances identiques au bare metal avec une empreinte mémoire minimale. C'est la méthode privilégiée pour les serveurs d'inférence Ollama ou vLLM en production.

```
# Configuration du conteneur LXC (dans /etc/pve/lxc/300.conf)
lxc.cgroup2.devices.allow: c 195:* rwm
lxc.cgroup2.devices.allow: c 236:* rwm
lxc.cgroup2.devices.allow: c 509:* rwm
lxc.mount.entry: /dev/nvidia0 dev/nvidia0 none bind,optional,create=file
lxc.mount.entry: /dev/nvidiactl dev/nvidiactl none bind,optional,create=file
lxc.mount.entry: /dev/nvidia-uvm dev/nvidia-uvm none bind,optional,create=file
lxc.mount.entry: /dev/nvidia-uvm-tools dev/nvidia-uvm-tools none bind,optional,create=file

# Le driver NVIDIA doit être installé sur l'hôte ET dans le conteneur (même version)
# Sur l'hôte :
apt install nvidia-driver-560 -y

# Dans le conteneur (installer uniquement les libs, pas le driver kernel)
apt install nvidia-utils-560 libnvidia-compute-560 -y

# Vérifier
lxc-attach -n 300 -- nvidia-smi
```

Ollama et vLLM sur Proxmox avec GPU

Déployer un serveur d'inférence LLM sur Proxmox avec GPU passthrough est une alternative crédible aux solutions cloud pour les organisations souhaitant garder le contrôle de leurs données. Les performances sont à 95-98% du bare metal pour l'inférence, ce qui est négligeable en pratique. Pour approfondir les architectures de déploiement LLM, consultez notre guide RAG en production.

```
# Déploiement Ollama dans un LXC avec GPU
# Dans le conteneur :
curl -fsSL https://ollama.ai/install.sh | sh

# Vérifier la détection GPU
ollama serve &
curl http://localhost:11434/api/tags

# Charger un modèle
ollama pull llama3.1:70b-instruct-q4_K_M

# Déploiement vLLM dans une VM avec GPU passthrough
pip install vllm
vllm serve meta-llama/Llama-3.1-70B-Instruct \
  --tensor-parallel-size 2 \
  --gpu-memory-utilization 0.90 \
  --max-model-len 32768
```

Benchmarks : passthrough vs bare metal

Test	Bare Metal	VM Passthrough	LXC Passthrough	Overhead
nvidia-smi power draw (idle)	25W	25W	25W	0%
CUDA bandwidthTest H2D	12.8 GB/s	12.6 GB/s	12.8 GB/s	VM: 1.5%, LXC: 0%
Llama 3.1 8B inference (tok/s)	142	138	141	VM: 2.8%, LXC: 0.7%
Llama 3.1 70B inference (tok/s)	24	23.5	23.8	VM: 2.1%, LXC: 0.8%
PyTorch MNIST training (s)	8.2	8.4	8.2	VM: 2.4%, LXC: 0%
Stable Diffusion XL (img/min)	18.5	18.1	18.4	VM: 2.2%, LXC: 0.5%

Troubleshooting

Problèmes courants et solutions

```
# 1. "IOMMU group is not viable" — Le GPU partage un groupe avec d'autres devices
# Solution : ACS override (dernier recours)
GRUB_CMDLINE_LINUX_DEFAULT="quiet intel_iommu=on iommu=pt
pcie_acs_override=downstream,multifunction"

# 2. "BAR 3: cannot reserve [mem]" — Conflit d'espace mémoire PCI
# Solution : augmenter l'espace MMIO
echo "options kvm ignore_msrs=1" >> /etc/modprobe.d/kvm.conf

# 3. VM freeze au démarrage avec GPU — ROM GPU manquante
# Solution : extraire et fournir la ROM
cd /sys/bus/pci/devices/0000:01:00.0/
echo 1 > rom
cat rom > /usr/share/kvm/gpu-rtx4090.rom
echo 0 > rom
# Dans la conf VM : hostpci0: 01:00,pcie=1,romfile=gpu-rtx4090.rom

# 4. Le GPU ne se réinitialise pas après arrêt VM (AMD reset bug)
# Solution : vendor-reset module (voir section AMD ci-dessus)

# 5. dmesg errors "vfio-pci: cannot reset device"
echo 1 > /sys/bus/pci/devices/0000:01:00.0/reset
```

FAQ — Questions fréquentes

Peut-on passer le même GPU à plusieurs VMs simultanément ?

Non, pas avec le passthrough classique. Un GPU physique ne peut être attribué qu'à une seule VM à la fois. Pour partager un GPU entre plusieurs VMs, il faut utiliser SR-IOV/MIG (GPU datacenter NVIDIA) ou une solution vGPU. Les GPU grand public (GeForce, Radeon) ne supportent pas le partage matériel. L'alternative pour les petits budgets est de passer le GPU à un conteneur LXC qui expose un service d'inférence via API (Ollama, vLLM), accessible par toutes les VMs du réseau.

Le passthrough GPU fonctionne-t-il avec les conteneurs LXC non privilégiés ?

Oui, mais cela nécessite une configuration supplémentaire des cgroups et des permissions sur les device nodes. Les conteneurs non privilégiés (par défaut dans Proxmox) remappent les UID/GID, ce qui complique l'accès aux fichiers de device /dev/nvidia*. La solution recommandée est d'utiliser un conteneur privilégié pour les workloads GPU, en acceptant le compromis de sécurité, ou de configurer des règles cgroup2 spécifiques avec les bons mappings UID.

Quel est l'impact du passthrough GPU sur la live migration ?

Une VM avec un GPU en passthrough ne peut pas être migrée à chaud (live migration). Le GPU est un périphérique physique attaché à un slot PCI spécifique du serveur hôte — il ne peut pas "suivre" la VM vers un autre nœud. La migration nécessite un arrêt de la VM, la migration, puis le rattachement à un GPU disponible sur le nœud de destination. Pour les workloads ML, cela implique de concevoir l'application pour gérer les interruptions (checkpointing des modèles en cours d'entraînement).

Ayi NEDJIMI Consultants — Expert cybersécurité offensive & intelligence artificielle

ayinedjimi-consultants.fr · ayi@ayinedjimi-consultants.fr