Proxmox VE - Retour d'expérience : Utilisation de périphériques physiques dans vos VMs (USB, HDD, GPU)

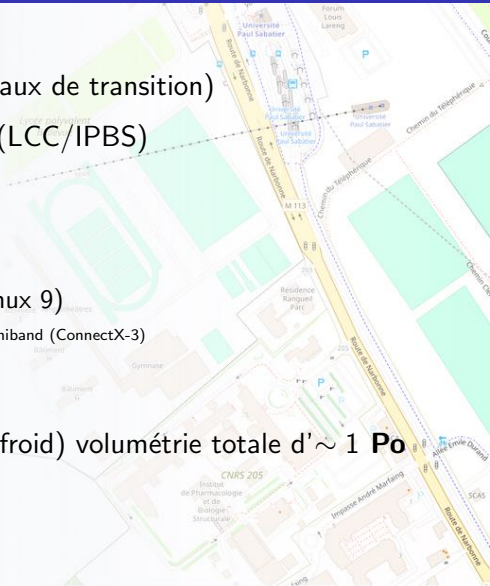
Jérôme COLOMBET

Josy Resinfo - GT Proxmox VE

8 octobre 2024

L'entité LCC - Contexte

- **L**aboratoire de **C**himie de **C**oordination (chimie métaux de transition)
- 3 salles serveurs réparties sur le campus CNRS 205 (LCC/IPBS)
- 4 clusters d'hyperviseurs Proxmox VE 8
  - 1 cluster - PROD
  - 1 cluster - Pre-PROD
  - 1 cluster - Tête HPC SLURM + 42 nœuds (Almalinux 9)
    - Passthrough HDD (dell md1200) & PCIe GPU (RTX A4000) & PCI Infiniband (ConnectX-3)
  - 1 cluster Proxmox VE - Campus 205
    - Passthrough USB dongle (Sentinel Key)
- 3 stockages centralisés ZFS [1] [2](production, backup, froid) volumétrie totale d'$\sim$ 1 **Po**



1. https://resinfo-gt.pages.in2p3.fr/zfs/doc/index.html
2. https://homepages.lcc-toulouse.fr/colombet/capitoul_retour_exp_zfs.pdf

Fonctionnement du Passthrough

- Pour un KVM, le passthrough est le moyen direct d'accéder aux matériels de l'hyperviseur
- Possible grâce à la technologie IOMMU [3] (Input-Output Memory Management Unit)
- IOMMU permet de mapper les périphériques physiques directement à la VM (KVM)
- Pour utiliser le passthrough avec Proxmox VE, les pré-requis suivants sont nécessaires :
  - **Firmware** : Le support IOMMU doit être activé dans le BIOS ou UEFI
  - **Matériel compatible** : Le processeur et la carte mère doivent supporter IOMMU
    - Pour Intel, c'est la technologie VT-d
    - Pour AMD, c'est AMD-Vi
    - Ajouter au fichier /etc/default/grub : GRUB_CMDLINE_LINUX_DEFAULT='intel_iommu=on **OU** amd_iommu=on"
- Et pour un CT ? c'est possible via la configuration des CT dans l'hyperviseur

3. https://en.wikipedia.org/wiki/Input-output_memory_management_unit

Passthrough USB dans un CT

1. Identifier le périphérique USB (idVendor et idProduct), forme Bus 001 Device 003 : ID 1234 :abcd

```
# lsusb
Bus 001 Device 003: ID 0403:c580 Future Technology Devices International, Ltd HID UNIKEY dongle

# ls -al /dev/bus/usb/001/00*
crw-rw-r-- 1 root root 189, 1 Sep 29 15:53 /dev/bus/usb/001/002
```

2. Déterminer l'ID du CT afin de paramétrer le périphérique USB à sa configuration

```
# pct list

# cat <<EOF >> /etc/pve/lxc/VMID.conf
lxc.cgroup2.devices.allow: c 189:* rwm
lxc.mount.entry = /dev/bus/usb/001/003 dev/bus/usb/001/003 none bind,optional,create=file
EOF
```

3. Redémarrer et vérifier dans le conteneur LXC

```
# pct restart VMID

# pct exec VMID lsusb
Bus 001 Device 003: ID 0403:c580 Future Technology Devices International, Ltd HID UNIKEY dongle
```

4. Règles udev pour la lisibilité / CT unprivileged ne pas oublier les permissions

Passthrough USB [4] dans un KVM

1. Identifier le périphérique USB (idVendor et idProduct), forme Bus 001 Device 003 : ID 1234 :abcd

```
# lsusb
Bus 001 Device 003: ID 0403:c580 Future Technology Devices International, Ltd HID UNIKEY dongle
Bus 001 Device 002: ID 04b9:0300 Rainbow Technologies, Inc. SafeNet USB SuperPro/UltraPro
```

2. Déterminer l'ID du KVM afin de paramétrer le périphérique USB à sa configuration

```
# qm list

# qm set VMID --usb0 host=0403:c580

# qm config VMID | grep usb
usb0: host=04b9:0300
usb1: host=0403:c580
```

3. Redémarrer et vérifier dans le KVM

```
# qm restart VMID
# qm monitor VMID
Entering QEMU Monitor for VM VMID type 'help' for help
qm> info usbhost
  Bus 1, Addr 3, Port 13, Speed 12 Mb/s
    Class 00: USB device 0403:c580, HID UNIKEY
  Bus 1, Addr 2, Port 12, Speed 12 Mb/s
    Class ff: USB device 04b9:0300, Sentinel HL
```

4. https://pve.proxmox.com/wiki/USB_Devices_in_Virtual_Machines

Passthrough HDD [5] dans un KVM - physique

1. Pour exporter un disque physique vers un KVM, s'assurer qu'il n'est pas utilisé par l'hyperviseur
2. Identifier le disque à passer en passthrough et déterminer l'ID de la VM cible

```
# ls -l /dev/disk/by-id/
lrwxrwxrwx 1 root root  9 May  1 12:48 <ID-DISQUE> -> ../../sdxx

# qm list
VMID    www.lcc-toulouse.fr   running      6144        60.00           2183375
```

3. Ajouter le disque physique en passthrough à votre VM

```
qm set VMID -virtio2 /dev/disk/by-id/<ID-DISQUE>

>> virtio2 fait est la position sur le bus VirtIO, mais vous pouvez le changer selon vos besoins
```

4. Privilégier le PCI passthrough des cartes contrôleurs si plusieurs disques à assigner
5. Limites disques : 4 x IDE, 14 x SCSI, 16 x VIRTIO et 6 x SATA
6. Autre limite, celle des ressources matérielles et capacités de l'hyperviseur

5. https://pve.proxmox.com/wiki/Passthrough_Physical_Disk_to_Virtual_Machine_(VM)

Passthrough HDD dans un KVM - protocole 9p [7] virtio, alternative au NFS

- Plan 9 Filesystem Protocol, est un système de fichiers réseau pour l'OS Plan 9 - Bell Labs
  - Sur Proxmox VE (debian), le protocole 9pfs [6] est actif dans le Kernel et dans Qemu
  - Les répertoires de l'hyperviseur sont rendu accessible à la VM via le FS virtuels (virtio-9p-device)
  - Il est possible de partager les répertoires à plusieurs VM simultanément
- Sur l'hyperviseur ajouter la configuration du KVM

```
# more /etc/pve/qemu-server/VMID.conf
...
args: -fsdev local,security_model=passthrough,id=fsdev0,path=<mount_tag> \
-device virtio-9p-pci,id=fs0,fsdev=fsdev0,mount_tag=datashare,bus=pci.0,addr=0x4
```

- Dans le KVM faire un montage dans le fstab

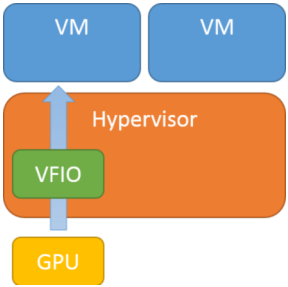
```
# more /etc/fstab
...
rep <mount_tag> 9p trans=virtio,version=9p2000.L,nobootwait,rw,_netdev,msize=262144 0 0
```

>> <mount_tag> est la balise associee par le serveur a chacun des points de montage exportes

6. https://wiki.qemu.org/Documentation/9psetup
7. https://www.kernel.org/doc/Documentation/filesystems/9p.txt

Passthrough PCI[8] - PCIe - GPU dans un KVM

1. Activer le module VFIO (Virtual Fonction I/O) sur l'hôte
   - Pour exposer l'accès direct au PCI
   - Les registres et les interruptions sont directement re-dirigés au KVM
   - Les accès DMA sont sécurisés par IOMMU
   - La carte PCI n'accède qu'à la zone de mémoire autorisée



http ://www.virtualopensystems.com/

2. Ajouter la carte PCI GPU passthrough à votre VM

```
# lspci | grep -i nvidia
05:00.0 VGA compatible controller: NVIDIA Corporation GA104GL [RTX A4000]

# qm set VMID -hostpci0 05:00.0
# qm config VMID | grep hostpci
hostpci0: 0000:05:00.0,pcie=1,x-vga=1
```

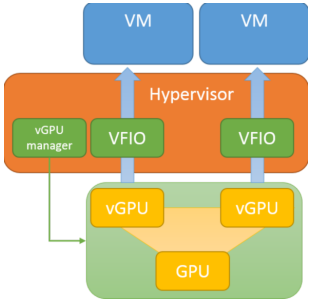
3. Installer le pilote natif du périphérique dans le système d'exploitation du KVM

```
# ./NVIDIA-Linux-x86_64-550.40.07.run --no-x-check --no-cc-version-check --dkms --accept-license
# nvidia-smi
NVIDIA-SMI 550.54.14        Driver Version: 550.54.14        CUDA Version: 12.4
```

8. https://pve.proxmox.com/wiki/PCI(e)_Passthrough

Passthrough vGPU NVIDIA [11] dans un KVM - partie 1

1. Ce n'est pas gratuit [9] mais via un compte DEV [10] il est possible d'évaluer la solution vGPU
2. Installer sur l'hyperviseur NVIDIA vGPU manager et les drivers vGPU hôte
3. Installer sur l'hyperviseur NVIDIA GRID licence pour débloquer la création des vGPU
4. Sélectionner le vGPU souhaité (vProfile, vCore, vMem) issu des propositions vGPU
5. Ajouter le vGPU (mediated device) en passthrough à votre VM et installer les pilotes pour l'OS de la VM



http ://www.virtualopensystems.com/

9. https://www.nvidia.com/fr-fr/data-center/buy-grid/
10. https://docs.nvidia.com/vgpu/
11. https://pve.proxmox.com/wiki/NVIDIA_vGPU_on_Proxmox_VE

Passthrough vGPU - partie 2 - exemple Nvidia P2000

```
# nvidia-smi
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 535.129.03             Driver Version: 535.129.03   CUDA Version: N/A      |
|-------------------------------+----------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute M. |
|                                         |                      |               MIG M. |
|   0  Quadro P2000                   On  | 00000000:2B:00.0 Off |                  N/A |
| 47%   37C    P8              7W /  75W |     23MiB /  5120MiB |      0%      Default |
|                                         |                      |                  N/A |
+-----------------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

```
# nvidia-smi vgpu
+---------------------------------------------------------------------------------+
| NVIDIA-SMI 535.129.03              Driver Version: 535.129.03                   |
|---------------------------------+------------------------------+----------------+
| GPU  Name                       | Bus-Id                       | GPU-Util       |
|      vGPU ID     Name           | VM ID     VM Name            | vGPU-Util      |
|=================================+==============================+================|
|   0  Quadro P2000               | 00000000:2B:00.0             |   0%           |
+---------------------------------+------------------------------+----------------+
```

Passthrough vGPU - partie 3 - exemple Nvidia P2000

```
# nvidia-smi vgpu -s
GPU 00000000:2B:00.0
    GRID P40-1Bx
    GRID P40-1Q
    GRID P40-2Q
    GRID P40-3Q
    GRID P40-4Q
    GRID P40-6Q
    GRID P40-8Q
    GRID P40-12Q
    GRID P40-24Q
    GRID P40-1A
    GRID P40-2A
    GRID P40-3A
    GRID P40-4A
    GRID P40-6A
    GRID P40-8A
    GRID P40-12A
    GRID P40-24A
    GRID P40-2B
    GRID P40-2B4
    GRID P40-1B4
```

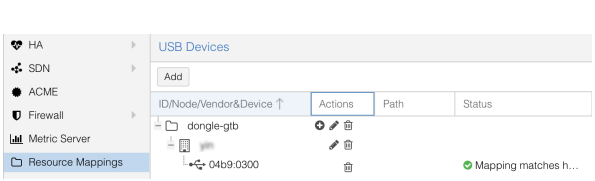
Votre GPU virtualisé, vous aurez 4 profiles distincts proposés

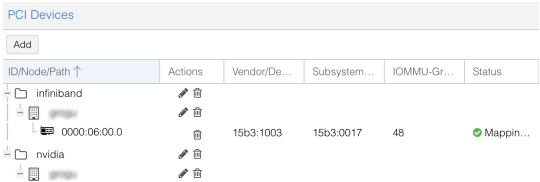
En fonction de vos besoins selectionner le profile pour votre usage (qt de mémoires, qt de coeurs) :

- **A** : Virtual Applications (vApps)
- **B** : Virtual Desktops (vPC)
- **C** : AI/Machine Learning/Training (vCS or vWS)
- **Q** : Virtual Workstations (vWS)

Proxmox VE - Mappings via API et Web UI

- Proxmox VE 8.0 [12], administration du mapping PCI et USB via l'API & WebUI
- Pas besoin de root pour donner accès à des périphériques spécifiques à un utilisateur
- En cluster, le mapping assure que le nœud cible dispose également d'un périphérique valide pour la migration d'un KVM utilisant le passthrough PCI ou USB
- Dans la configuration VMID.conf = `hostpci0 : mapping=nvidia,pcie=1,x-vga=1`
- Attention si HA ou migration, il faut le même matériel sur les autres noeuds du cluster





12. `https://pve.proxmox.com/wiki/Roadmap#Proxmox_VE_8.0`

Conclusion

1. Avantages :
   - Accès direct aux ressources matérielles
   - Performances identique à l'hyperviseur
   - Utilisation optimale du matériel physique (vGPU)
   - Facilité de mise en œuvre avec la WebUI depuis la v8

2. Pré-requis (inconvénients) :
   - Complexité de configuration (IOMMU, ...)
   - Vérifier la prise en charge du matériel pour le passthrough
   - Partage du matériel limité entre les VM
   - Risque d'instabilité (problèmes pilotes ou mauvaise configuration)

Questions

Liste ESR :
virtualisation@services.cnrs.fr

GT Proxmox VE :
proxmox-gt@listes.resinfo.org

: jerome.colombet@lcc-toulouse.fr
: https://homepages.lcc-toulouse.fr/colombet/