ATTENTION CETTE DOCUMENTATION EST EN COURS DE REDACTION

GPU Passthrough avec QEMU

Attention, ce sujet est bien trop complexe, impossible d'être exhaustif.

Prérequis

Hardware :

- Une carte mère supportant le choix de la carte graphique par défaut
- Un CPU compatible VT-d
- Deux cartes graphiques dans votre PC (exemple : votre iGPU et votre GPU NVIDIA)
- un Dummy Plug

Software :

- qemu en version 4.0 ou supperieur, sinon voir ceci.
- ovmf
- Looking Glass
- Un iso de Windows 10
- l'iso des pilotes VirtIO

Autre :

- Une maîtrise des systèmes Windows et Linux.
- Des compétences/connaissance avec qemu, grub/systemd-boot
- Du temps et de la patience. Vraiment!

Ma configuration :

Voici dans quels conditions ont été réalisé cette page :

- Carte mère: ASUS Z170 Gaming
- CPU: Intel i5 7600K
- RAM: 16Go
- GPU: Nvidia GEFORCE 1070
- Stockage : SSD NVMe pour l'host, SSD SATA pour le guest
- OS: Archlinux

Configuration de l'Host

En partant du principe que votre OS est installer.

Il faut commencer par aller dans votre BIOS/EFI, et activer la virtualisation, la technologie VT-d et définir l'iGPU comme carte graphique principale.

Ensuite branchez votre Dummy plug sur votre GPU, et vos écrans sur votre iGPU.

Ensuite configurer votre GRUB en ajoutant l'option **intel_iommu=on** a votre noyaux.

Ensuite il faut demander a votre initramfs de charger les pilotes vfio en ajoutant les modules **vfio_pci**, **vfio, vfio_iommu_type1** et **vfio_virqfd** dans le fichier */etc/mkinitcpio.conf*.

Recharger vos configurations :

```
# mkinitcpio -P && grub-mkconfig -o <EMPLACEMENT DE VOTRE grub.cfg>
```

Redémarrez, et lancer la commande :

```
# for iommu_group in $(find /sys/kernel/iommu_groups/ -maxdepth 1 -mindepth 1 -type d);do echo "IOMMU group $(basename "$iommu_group")"; for device in $(\ls -1 "$iommu_group"/devices/); do if [[ -e "$iommu_group"/devices/"$device"/reset ]]; then echo -n "[RESET]"; fi; echo -n $'\t';lspci -nns "$device"; done; done
```

vous devriez optenir quelque chose comme ceci :

```
IOMMU group 7
[RESET] 00:1c.0 PCI bridge [0604]: Intel Corporation 100 Series/C230 Series Chipset Family PCI Express Root Port #1 [8086:a110] (rev f1)
IOMMU group 5
    00:17.0 SATA controller [0106]: Intel Corporation Q170/Q150/B150/H170/H110/Z170/CM236 Chipset SATA Controller [AHCI Mode] [8086:a102] (rev 31)
IOMMU group 3
    00:14.0 USB controller [0c03]: Intel Corporation 100 Series/C230 Series Chipset Family USB 3.0 xHCI Controller [8086:a12f] (rev 31)
IOMMU group 11
[RESET] 03:00.0 USB controller [0c03]: ASMedia Technology Inc. ASM1142 USB 3.1 Host Controller [1b21:1242]
IOMMU group 1
    00:01.0 PCI bridge [0604]: Intel Corporation Xeon E3-1200 v5/E3-1500 v5/6th Gen Core Processor PCIe Controller (x16) [8086:1901] (rev 05)
[RESET] 01:00.0 VGA compatible controller [0300]: NVIDIA Corporation GP104 [GeForce GTX 1070] [10de:1b81] (rev a1)
    01:00.1 Audio device [0403]: NVIDIA Corporation GP104 High Definition Audio Controller [10de:10f0] (rev a1)
IOMMU group 8
[RESET] 00:1d.0 PCI bridge [0604]: Intel Corporation 100 Series/C230 Series
```

```
Chipset Family PCI Express Root Port #9 [8086:a118] (rev f1)
IOMMU group 6
[RESET] 00:1b.0 PCI bridge [0604]: Intel Corporation 100 Series/C230 Series
Chipset Family PCI Express Root Port #17 [8086:a167] (rev f1)
IOMMU group 4
    00:16.0 Communication controller [0780]: Intel Corporation 100
Series/C230 Series Chipset Family MEI Controller #1 [8086:a13a] (rev 31)
IOMMU group 12
[RESET] 04:00.0 Non-Volatile memory controller [0108]: Samsung Electronics
Co Ltd NVMe SSD Controller SM981/PM981 [144d:a808]
IOMMU group 2
[RESET] 00:02.0 VGA compatible controller [0300]: Intel Corporation HD
Graphics 630 [8086:5912] (rev 04)
IOMMU group 10
[RESET] 00:1f.6 Ethernet controller [0200]: Intel Corporation Ethernet
Connection (2) I219-V [8086:15b8] (rev 31)
IOMMU group 0
    00:00.0 Host bridge [0600]: Intel Corporation Xeon E3-1200 v6/7th Gen
Core Processor Host Bridge/DRAM Registers [8086:591f] (rev 05)
IOMMU group 9
    00:1f.0 ISA bridge [0601]: Intel Corporation Z170 Chipset LPC/eSPI
Controller [8086:a145] (rev 31)
    00:1f.2 Memory controller [0580]: Intel Corporation 100 Series/C230
Series Chipset Family Power Management Controller [8086:a121] (rev 31)
    00:1f.3 Audio device [0403]: Intel Corporation 100 Series/C230 Series
Chipset Family HD Audio Controller [8086:a170] (rev 31)
    00:1f.4 SMBus [0c05]: Intel Corporation 100 Series/C230 Series Chipset
Family SMBus [8086:a123] (rev 31)
```

Et il vous faudra identifier votre GPU et de noter des valeurs pour tard, pour moi c'est :

```
01:00.0 - 10de:1b81 (GPU)
01:00.1 - 10de:10f0 (Audio)
```

Ensuite faut définir au noyau quel carte graphique on reserve a la VM :

/etc/modprobe.d/gpu-passthrough.conf

```
blacklist nouveau
blacklist nvidia
options kvm ignore_msrs=1 # Pour éviter les crashs de certaines app
dans le guest
options vfio-pci ids=10de:1b81,10de:10f0 disable_vga=1 # Adaptez avec
vos valeurs :)
```

Redémarrez et tapez la commande :

```
# lspci -k
```

afin de vérifier l'utilisation du module vfio-pci :

```
....
01:00.0 VGA compatible controller: NVIDIA Corporation GP104 [GeForce GTX
1070] (rev a1)
    Subsystem: ASUSTeK Computer Inc. GP104 [GeForce GTX 1070]
 Kernel driver in use: vfio-pci
    Kernel modules: nouveau
01:00.1 Audio device: NVIDIA Corporation GP104 High Definition Audio
Controller (rev a1)
    Subsystem: ASUSTeK Computer Inc. GP104 High Definition Audio Controller
 Kernel driver in use: vfio-pci
    Kernel modules: snd_hda_intel
....
```

Configuration du Guest

Démarrez votre VM sans la carte graphique, et tout les modules necessaires. Par exemple avec la commande :

```
qemu-system-x86_64 -L . -drive
if=pflash,format=raw,readonly,file=/usr/share/ovmf/x64/OVMF_CODE.fd -drive
if=pflash,format=raw,file=/usr/share/ovmf/x64/OVMF_VARS.fd -enable-kvm  -m
8G -drive file=/dev/sda,format=raw,if=virtio -netdev
type=user,id=net0,hostfwd=tcp::3389-:3389 -device virtio-net-pci,netdev=net0
-soundhw hda -cpu
'host,hv_relaxed,hv_spinlocks=0x1fff,hv_vapic,hv_time,hv_vendor_id=whatever1
23,+kvm_pv_unhalt,+kvm_pv_eoi,kvm=off' -smp 2,sockets=1,cores=2,threads=1  -
device ivshmem-plain,memdev=ivshmem -object memory-backend-
file,id=ivshmem,share=on,mem-path=/dev/shm/looking-glass,size=32M -spice
port=5900,addr=127.0.0.1,disable-ticketing -device virtio-serial-pci -device
virtserialport,chardev=spicechannel0,name=com.redhat.spice.0 -chardev
spicevmc,id=spicechannel0,name=vdagent -chardev
socket,path=/tmp/qga.sock,server,nowait,id=qga0 -device virtio-serial -
device virtserialport,chardev=qga0,name=org.qemu.guest_agent.0 -monitor
telnet:127.0.0.1:55555,server,nowait -device virtio-keyboard-pci -audiodev
pa,id=pa1,server=/run/user/1000/pulse/native -drive
file=win10.iso,index=0,media=cdrom -drive file=virtio-
win-0.1.141.iso,index=1,media=cdrom -vga std -display sdl
```

Installez votre Windows 10. Je vous conseille :

- de mettre un mot de passe a votre session (Attention a bien mettre l'autologin !!!)
- Activer RDP
- D'installer tout les pilotes/updates disponibles

Une fois fait, télécharger (sur votre guest) les pilotes spécific Redhat et installez tout les pilotes de l'archive (Voir ceci pour gagner du temps)

Installez aussi Visual C++.

Une fois fait, télécharger la dernieres release de Looking Glass pour Windows, placer l'executable dans un endroit controllé, et ajoutez une clé de registre pour le lancer au démarrage (HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run).

Eteignez la VM et lancez la avec votre carte graphique et sans l'affichage virtuel, en ajustant les bons argument, par exemple :

```
# qemu-system-x86_64 [....] -vga none -display none -device vfio-
pci,host=01:00.0,multifunction=true -device vfio-pci,host=01:00.1
```

Et normalement, en lancant Looking Glass, vous devriez récupérer l'affichage de votre VM.