Table des matières

Install et config de qemu 3
/etc/synergy.conf 4
Config de la VM 5
Backup de la VM 6
Infos VM 7

Le but de cette doc est de voir comment on peut utiliser la virtu sous Linux en dédiant une carte graphique à une VM (VGA/PCI passthrough).



Pré-requis indispensables :

→ un processeur supportant le VT-d ;
→ une carte mère supportant également le VT-d ;
→ 2 cartes graphiques. Le chipset graphique intégré aux cores i5/i7 est suffisant pour la partie Linux (voir patch i915 cependant) ;
→ 2 écrans ou un écran avec plusieurs entrées.

Ci-dessous le hardware/software utilisé dans ce guide :

Hardware

	MB	CPU	RAM	GPU	Disk
host	Asus Z97-AR	Core i5-4460 @ 3.20 Ghz	16 Gb	Radeon R240	SSD 840 EVO 250G
guest	ICH9	Core i5-4460 @ 3.20 Ghz	10 Gb	GTX 660	LVM

Software

	OS	Kernel	Virt	Nvidia	Syngergy
host	Debian Jessie 8.1	3.16.7-ckt11-1+deb8u5	qemu 2.1.2	n/a	1.7.4 (serveur)
guest	Windows 10 Pro 64 bits	n/a	n/a	358.91	1.7.4 (client)



Le patch i915 est indispensable si on utilise un chipset style Intel HD 4600. Sans ce patch il n'y a pas d'arbitrage VGA et donc la VM ne peut pas accéder au GPU dédié de la VM. On peut malgré tout se passer de ce patch en utilisant OVMF (cf. https://wiki.archlinux.org/index.php/PCI_passthrough_via_OVMF).

Voir les liens ci-dessous pour l'install du patch :

- http://unix.ndlp.info/doku.php/informatique:nix:linux:qemu:vga_passthrough_i915
- https://lkml.org/lkml/2014/5/9/517

Si vous utilisez une carte graphique dédiée pour le host, pas besoin de ce patch.

Install et config de qemu

```
apt-get install qemu seabios
```

Pour avoir une version plus récente de qemu (2.6.2) :

qemu_2.6.2-1_amd64.deb

- package pas encore testé.

- Récupérer Synergy ici (https://synergy-project.org/nightly) et l'installer
- Modifier le fichier **/etc/default/grub** :

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet intel_iommu=on vfio_iommu_type1.allow_unsafe_interrupts=1"
```

- Mise à jour de grub :

```
update-grub
```

- Modifier le fichier **/etc/modules** :

```
# /etc/modules: kernel modules to load at boot time.
#
# This file contains the names of kernel modules that should be loaded
# at boot time, one per line. Lines beginning with "#" are ignored.

pci_stub
vfio
vfio_iommu_type1
vfio_pci
kvm
kvm_intel
```

```
cat /etc/modprobe.d/blacklist.conf
```

```
blacklist nouveau
```

- On détecte ensuite les IDs du GPU nvidia :

```
root@ben-pc:~# lspci -nn | grep NVIDIA
01:00.0 VGA compatible controller [0300]: NVIDIA Corporation GK106 [GeForce GTX 660] [10de:11c0] (rev a1)
01:00.1 Audio device [0403]: NVIDIA Corporation GK106 HDMI Audio Controller [10de:0e0b] (rev a1)
```

```
root@ben-pc:~# lspci -n |egrep "01.00.[0-1]"
01:00.0 0300: 10de:11c0 (rev a1)
01:00.1 0403: 10de:0e0b (rev a1)
```

- Modifier le fichier **/etc/initramfs-tools/modules** :

```
pci_stub ids=10de:11c0,10de:0e0b
```

- Mise à jour de l'initrd :

```
update-initramfs -k all -u
```

- Créer le fichier **/etc/vfio-pci1.cfg** :

```
0000:01:00.0
0000:01:00.1
```

- Créer un bridge réseau :

```
apt-get install remove network-manager
```

- Modifier le fichier **/etc/interfaces** :

```
auto lo
iface lo inet loopback

iface eth0 inet manual

auto br0
iface br0 inet dhcp
    bridge_ports eth0
```

- Rebooter
- Checker que les devices en question ont bien été pris en compte :

```
root@ben-pc:~# dmesg | grep pci-stub
[    1.483508] pci-stub: add 10DE:11C0 sub=FFFFFFFF:FFFFFFFF cls=00000000/00000000
[    1.483517] pci-stub 0000:01:00.0: claimed by stub
[    1.483522] pci-stub: add 10DE:0E0B sub=FFFFFFFF:FFFFFFFF cls=00000000/00000000
[    1.483526] pci-stub 0000:01:00.1: claimed by stub
```

/etc/synergy.conf

```
section: screens
    win-vm:
        halfDuplexCapsLock = false
        halfDuplexNumLock = false
        halfDuplexScrollLock = false
        xtestIsXineramaUnaware = false
        switchCorners = none
        switchCornerSize = 0
    ben-pc:
        halfDuplexCapsLock = false
        halfDuplexNumLock = false
        halfDuplexScrollLock = false
        xtestIsXineramaUnaware = false
        switchCorners = none
        switchCornerSize = 0
end

section: aliases
end

section: links
    win-vm:
        left = ben-pc
    ben-pc:
        right = win-vm
```

```
end

section: options
    relativeMouseMoves = false
    screenSaverSync = false
    win32KeepForeground = false
    switchCorners = none
    switchCornerSize = 0
    mousebutton(6) = keystroke(WWWBack) ;
    mousebutton(7) = keystroke(WWWForward) ;
end
```

Config de la VM

Il faut récupérer les drivers virtio pour windows ici : https://fedorapeople.org/groups/virt/virtio-win/direct-downloads/stable-virtio/virtio-win.iso (meilleures perfs qu'en IDE ou SCSI). Lors de l'install de Windows il faudra aller chercher ces drivers spécifiques pour accéder aux disques.

Ici LVM est utilisé pour créer les disques de la VM :

```
  lv_jeux2     datavg  -wi-ao---- 100.00g
  lv_jeux      vmvg    -wi-ao---- 120.00g
  lv_win10     vmvg    -wi-ao----  30.00g
```

Lors du premier lancement (pour l'install) on ne dispose pas encore de la souris ni du clavier. On lance tout simplement qemu en mode fenêtré, cf. ligne :

win10_install.sh

```
#!/bin/bash

configfile=/etc/vfio-pci1.cfg

vfiobind() {
    dev="$1"
        vendor=$(cat /sys/bus/pci/devices/$dev/vendor)
        device=$(cat /sys/bus/pci/devices/$dev/device)
        if [ -e /sys/bus/pci/devices/$dev/driver ]; then
                echo $dev > /sys/bus/pci/devices/$dev/driver/unbind
        fi
        echo $vendor $device > /sys/bus/pci/drivers/vfio-pci/new_id

}

modprobe vfio-pci

cat $configfile | while read line;do
    echo $line | grep ^# >/dev/null 2>&1 && continue
        vfiobind $line
done

qemu-system-x86_64 -enable-kvm -M q35 -m 10240 -cpu host \
-smp 4,sockets=1,cores=4,threads=1 \
-bios /usr/share/seabios/bios.bin -vga cirrus \
-net nic -net tap \
-device ioh3420,bus=pcie.0,addr=1c.0,multifunction=on,port=1,chassis=1,id=root.1 \
-drive file=/dev/vmvg/lv_win10,if=none,id=drive-virtio-disk0,format=raw \
-device virtio-blk-pci,scsi=off,addr=0x7,drive=drive-virtio-disk0,id=virtio-disk0,bootindex=1 \
-drive file=/home/ben/nas/softs/Windows10_pro.iso,id=isocd -device ide-cd,bus=ide.1,drive=isocd \
-drive file=/home/ben/nas/softs/virtio-win.iso,id=isocd2 -device ide-cd,bus=ide.2,drive=isocd2 \
-boot menu=on

exit 0
```

Une fois l'install ok, on binde la carte graphique, on désactive le mode fenêtré et on ajoute les disques que l'on souhaite :

win10.sh

```
#!/bin/bash

configfile=/etc/vfio-pci1.cfg

vfiobind() {
```

```
    dev="$1"
        vendor=$(cat /sys/bus/pci/devices/$dev/vendor)
        device=$(cat /sys/bus/pci/devices/$dev/device)
        if [ -e /sys/bus/pci/devices/$dev/driver ]; then
                echo $dev > /sys/bus/pci/devices/$dev/driver/unbind
        fi
        echo $vendor $device > /sys/bus/pci/drivers/vfio-pci/new_id

}

modprobe vfio-pci

cat $configfile | while read line;do
    echo $line | grep ^# >/dev/null 2>&1 && continue
        vfiobind $line
done

echo "Starting Synergy"
synergys --daemon --config /etc/synergy.conf

echo "Starting Samba"
/etc/init.d/samba start
echo

echo "Starting VM ..."
qemu-system-x86_64 -enable-kvm -M q35 -m 10240 -cpu host,kvm=off,hv-time=off,hv-relaxed=off,hv-vapic=off \
-rtc base=localtime \
-smp 4,sockets=1,cores=4,threads=1 \
-bios /usr/share/seabios/bios.bin -vga none \
-nographic \
-net nic -net tap \
-device ioh3420,bus=pcie.0,addr=1c.0,multifunction=on,port=1,chassis=1,id=root.1 \
-device vfio-pci,host=01:00.0,bus=root.1,addr=00.0,multifunction=on,x-vga=on \
-drive file=/dev/vmvg/lv_win10,if=none,cache=directsync,aio=native,id=drive-virtio-disk0,format=raw \
-device virtio-blk-pci,scsi=off,addr=0x7,drive=drive-virtio-disk0,id=virtio-disk0,bootindex=1 \
-drive file=/dev/vmvg/lv_jeux,if=none,cache=directsync,aio=native,id=drive-virtio-disk1,format=raw \
-device virtio-blk-pci,scsi=off,addr=0x8,drive=drive-virtio-disk1,id=virtio-disk1,x-data-plane=on \
-drive file=/dev/datavg/lv_jeux2,if=none,cache=directsync,aio=native,id=drive-virtio-disk2,format=raw \
-device virtio-blk-pci,scsi=off,addr=0x9,drive=drive-virtio-disk2,id=virtio-disk2,x-data-plane=on \
-drive file=/home/ben/nas/softs/Windows10_pro.iso,id=isocd -device ide-cd,bus=ide.1,drive=isocd \
-drive file=/home/ben/nas/softs/virtio-win.iso,id=isocd2 -device ide-cd,bus=ide.2,drive=isocd2 \
-soundhw hda \
-boot menu=on

echo "Closing VM ..."
echo
echo "VM closed"

echo "Stopping Synergy"
killall synergys

echo "Stopping Samba"
/etc/init.d/samba stop

exit 0
```

- Une fois l'install de Windows terminée, il faut installer le client Synergy sur le guest pour accéder à la souris et au clavier.

On peut aussi faire du passthrough sur des devices usb :

```
-device nec-usb-xhci \
-device usb-host,vendorid=0x413c,productid=0x2105 \
-device usb-host,vendorid=0x046d,productid=0xc01e
```

Backup de la VM

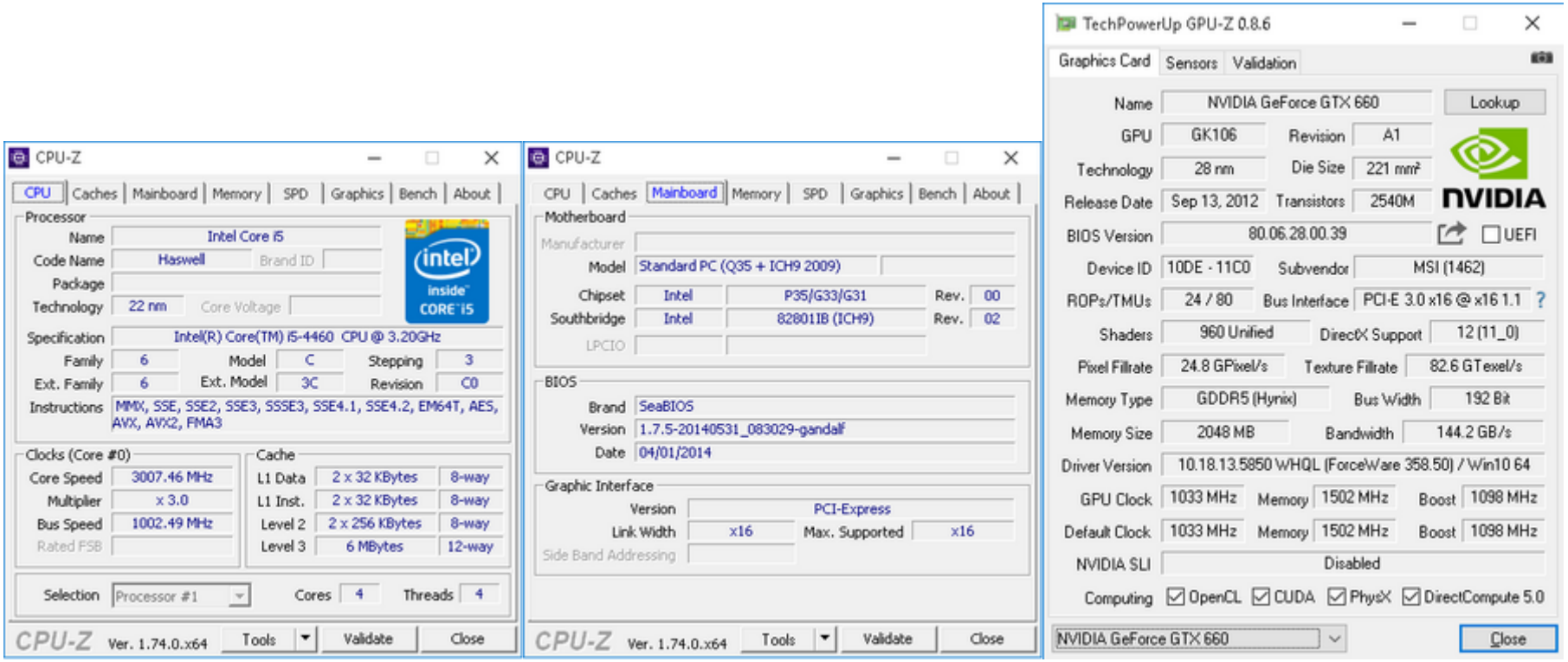
```
dd if=/dev/vmvg/lv_win10 bs=64k conv=sync of=${LOCAL_PATH}/lv_win.10img bs=64k

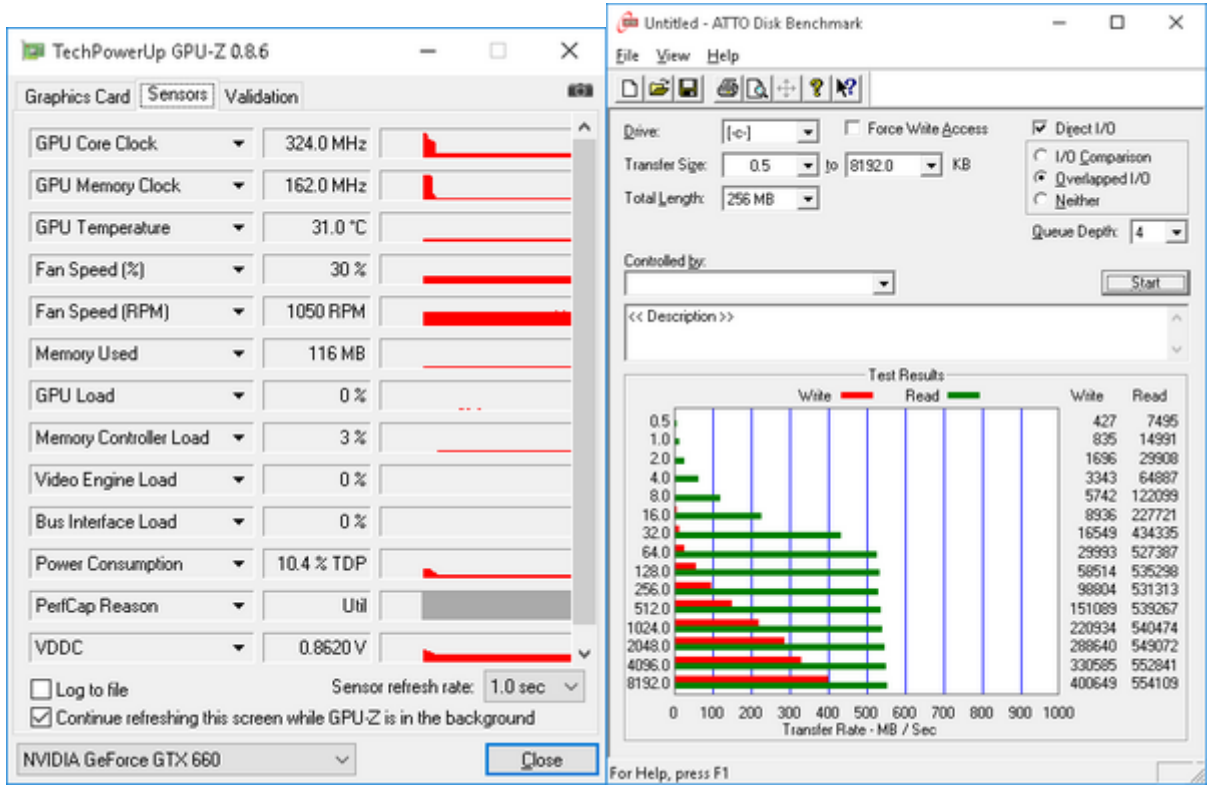
kpartx -a /dev/vmvg/lv_jeux
sleep 5
mount /dev/mapper/vmvg-lv_jeux1 /mnt -o ro
rsync -aur --delete --progress --exclude="pagefile.sys" /mnt/ ${LOCAL_PATH}/lv_jeux/
umount /mnt
```

```
sleep 5
kpartx -d /dev/vmvg/lv_jeux

kpartx -a /dev/datavg/lv_jeux2
sleep 5
mount /dev/mapper/datavg-lv_jeux2p1 /mnt -o ro
rsync -aur --delete --progress  /mnt/ ${LOCAL_PATH}/lv_jeux2/
umount /mnt
sleep 5
kpartx -d /dev/datavg/lv_jeux2
```

Infos VM




TechPowerUp GPU-Z 0.8.6
Graphics Card
Sensors
Validation
GPU Core Clock 324.0 MHz
GPU Memory Clock 162.0 MHz
GPU Temperature 31.0 °C
Fan Speed (%) 30 %
Fan Speed (RPM) 1050 RPM
Memory Used 116 MB
GPU Load 0 %
Memory Controller Load 3 %
Video Engine Load 0 %
Bus Interface Load 0 %
Power Consumption 10.4 % TDP
PerfCap Reason Util
VDDC 0.8620 V
Log to file
Sensor refresh rate: 1.0 sec
Continue refreshing this screen while GPU-Z is in the background
NVIDIA GeForce GTX 660
Close
Untitled - ATTO Disk Benchmark
File View Help
Drive: [-c-]
Force Write Access
Direct I/O
Transfer Size: 0.5 to 8192.0 KB
I/O Comparison
Overlapped I/O
Neither
Total Length: 256 MB
Queue Depth: 4
Controlled by:
Start
<< Description >>
Test Results
Write Read
0.5 427 7495
1.0 835 14991
2.0 1696 29908
4.0 3343 64887
8.0 5742 122099
16.0 8936 227721
32.0 16549 434335
64.0 29993 527387
128.0 58514 535298
256.0 98804 531313
512.0 151089 539267
1024.0 220934 540474
2048.0 288640 549072
4096.0 330585 552841
8192.0 400649 554109
0 100 200 300 400 500 600 700 800 900 1000
Transfer Rate - MB / Sec
For Help, press F1


NVIDIA GeForce Experience
Jeux
Pilotes
Mon équipement
SHIELD
Préférences
ShadowPlay
Connexion
WIN-VM
Processeur graphique : GeForce GTX 660
Processeur : Intel(R) Core(TM) i5-4460 CPU @ 3.20GHz
Mémoire : 10,00 Go de RAM (10,00 Go utilisables)
Résolution actuelle : 1920 x 1200, 32Hz
Version du pilote : 358.91
Système d'exploitation : Microsoft Windows 10 Professionnel
Vue d'ensemble
Optimisation de jeux
GameStream
ShadowPlay
Visualisation LED
Optimisation de jeux
Optimisez les paramètres de vos jeux PC
Prêt
GameStream
Lisez des jeux en streaming sur NVIDIA SHIELD depuis votre PC
Prêt
ShadowPlay
Enregistrez vos meilleures sessions de jeu !
Prêt
Visualisation LED
Configurez des effets lumineux pour votre machine de jeu !
Pas prêt
Envoyer un commentaire

A tester

```
QEMU_PA_SAMPLES=6144 QEMU_AUDIO_DRV=pa \
qemu-system-x86_64 -enable-kvm -m 8192 -cpu host,kvm=off \
-smp 4,sockets=1,cores=4,threads=1 \
-machine q35,accel=kvm \
-soundhw hda \
-device ioh3420,bus=pcie.0,addr=1c.0,multifunction=on,port=1,chassis=1,id=root.1 \
-device vfio-pci,host=$DEVICE1,bus=root.1,addr=00.0,multifunction=on,x-vga=on \
-device vfio-pci,host=$DEVICE2,bus=root.1,addr=00.1 \
-vga none \
-bios /usr/share/seabios/bios.bin \
-device virtio-net-pci,netdev=user.0,mac=52:54:00:03:02:01 \
-netdev user,id=user.0 \
-drive file=win7-x64_system.qcow2,if=none,id=drive-virtio-disk0,format=qcow2 \
-device virtio-blk-pci,scsi=off,addr=0x7,drive=drive-virtio-disk0,id=virtio-disk0,bootindex=1 \
-drive file=win7-games.qcow2,if=none,id=drive-virtio-disk1,format=qcow2 \
-device virtio-blk-pci,scsi=off,addr=0x8,drive=drive-virtio-disk1,id=virtio-disk1 \
-rtc base=localtime,driftfix=slew \
-device qxl \
-device usb-kbd \
-usbdevice host:1e7d:2d51
```

From:
https://unix-bck.ndlp.info/ - **Where there is a shell, there is a way**

Permanent link:
https://unix-bck.ndlp.info/doku.php/informatique:nix:linux:qemu:vga_passthrough

Last update: **2017/10/08 17:58**