Es posible asignar a una máquina virtual una tarjeta PCI o PCIe insertada en una ranura del host Proxmox. En el mundillo se conoce como PCI passthrough. En este artículo sólo cubriré la asignación de tarjetas PCI o PCIe (PCI Express) que no sean gráficas. Si queréis pasar tarjetas gráficas tenéis información sobre como hacerlo aquí.

Para poder pasar tarjetas PCI o PCIe a una máquina virtual en un host ProxmoxVE debes cumplir los siguientes requisitos:

REQUISITO 1: PROCESADOR CON EXTENSIONES DE VIRTUALIZACIÓN

Debes saber si tu procesador cuenta con las extensiones de virtualización. En el caso de tener un procesador Intel deberías contar tanto con la tecnología **VT-x** como con **VT-d**. En el caso de AMD necesitas tanto **AMD-v** como **AMD-Vi** . Pare ver si cuentas con dichas tecnologías ejecuta:

```
egrep -q '^flags.*(svm|vmx)' /proc/cpuinfo && echo Extensiones de virtualización disponibles
```

Lógicamente, si te sale el mensaje:

```
Extensiones de virtualización disponibles
```

...después de ejecutar el comando anterior, tu procesador cuenta con ellas. Si no te sale nada, olvídate de PCI Passthrough hasta comprarte un procesador decente.

Llegado el caso, si quieres saber más acerca de cuáles tecnologías soporta tu procesador (además de las de virtualización) puedes ejecutar:

Si tienes un procesador AMD:

```
grep -E --color svm /proc/cpuinfo
```

Si tienes un procesador Intel:

```
grep -E --color vmx /proc/cpuinfo
```

De esa forma te aparecerán resaltadas en color las tecnologías de virtualización y además se mostrarán todas las otras tecnologías (flags) soportadas.

NOTA: Algunos procesadores tienen soporte para IOMMU pero si el chipset no cuenta con el soporte, o no está activado en la BIOS, da igual que el procesador lo tenga porque el host no será capaz de usarlo.

Si cumples el requisito 1 pasa al

REQUISITO 2: SOPORTE PARA IOMMU

Debes saber si tienes soporte IOMMU en el sistema. Para ello, lógicamente, tienes que activarlo en la BIOS. Una vez activado, y para asegurarte 100% ejecuta:

```
dmesg | grep -e DMAR -e IOMMU
```

Si tienes un procesador Intel y la salida es algo como esto:

```
[ 9.344313] AMD IOMMUv2 driver by Joerg Roedel <jroedel@suse.de>
[ 9.344315] AMD IOMMUv2 functionality not available on this system
```

Es decir, si no sale nada de IOMMUv1 y de IOMMUv2, después de mostrarte la línea de Joerg Roedel te sale otra línea que pone que la funcionalidad no está disponible, no tienes soporte ni para IOMMUv1 ni para IOMMUv2. Ello teniendo en cuenta que no te hayas olvidado de

activarlo en la BIOS, claro.

Si por el contrario la salida es algo como esto:

```
[ 0.000000] ACPI: DMAR 0x00000000D5DE4A80 0003B4 (v01 HP ProLiant 00000001 \xffffffd2? 0000162E)
[ 0.015239] DMAR: Host address width 39
[ 0.015240] DMAR: DRHD base: 0x000000fed90000 flags: 0x1
[ 0.015244] DMAR: dmar0: reg_base_addr fed90000 ver 1:0 cap c9008020660262 ecap f010da
[ 0.015244] DMAR: RMRR base: 0x000000d5ffd000 end: 0x000000d5ffffff
[ 0.015245] DMAR: RMRR base: 0x000000d5ff6000 end: 0x000000d5ffcfff
[ 0.015246] DMAR: RMRR base: 0x000000d5f93000 end: 0x000000d5f94fff
[ 0.015247] DMAR: RMRR base: 0x000000d5f8f000 end: 0x000000d5f92fff
[ 0.015247] DMAR: RMRR base: 0x000000d5f7f000 end: 0x000000d5f8efff
[ 0.015248] DMAR: RMRR base: 0x000000d5f7e000 end: 0x000000d5f7efff
[ 0.015248] DMAR: RMRR base: 0x000000000f4000 end: 0x000000000f4fff
[ 0.015250] DMAR: RMRR base: 0x000000000e8000 end: 0x000000000e8fff
[ 0.015250] DMAR: RMRR base: 0x000000d5dee000 end: 0x000000d5deefff
[ 0.015252] DMAR-IR: IOAPIC id 8 under DRHD base 0xfed90000 IOMMU 0
[ 0.015252] DMAR-IR: HPET id 0 under DRHD base 0xfed90000
[ 0.015253] DMAR-IR: x2apic is disabled because BIOS sets x2apic opt out bit.
[ 0.015254] DMAR-IR: Use 'intremap=no_x2apic_optout' to override the BIOS setting.
[ 0.015404] DMAR-IR: Enabled IRQ remapping in xapic mode
[ 1.076836] AMD IOMMUv2 driver by Joerg Roedel <jroedel@suse.de>
[ 1.076836] AMD IOMMUv2 functionality not available on this system
```

Es decir, que se hace referencia a DMAR, aunque haya una línea que diga que la funcionalidad de IOMMUv2 no está disponible, significa que tienes soporte, aunque no tengas IOMMUv2.

NOTA 1: Aunque el chipset de tu placa base soporte IOMMU, la BIOS también tiene que tener programado el soporte a través de las tablas ACPI IVRS. Ambos son requisitos. No vale con sólo tener soporte en el chipset pero tener una BIOS pedorra.

NOTA 2: IOMMUv2 fue diseñado principalmente para las APUs con gráficos integrados como parte del HSA de AMD. A menos que estés tratando de pasar una tarjeta gráfica de una APU AMD (GPGPU) no necesitas IOMMUv2.

Si tienes un procesador AMD y la salida es algo como esto:

```
[ 9.344313] AMD IOMMUv2 driver by Joerg Roedel <jroedel@suse.de>
[ 9.344315] AMD IOMMUv2 functionality not available on this system
```

Es decir, si no sale nada de «Found IOMMU at ...» y después de mostrarte la línea de IOMMUv2 de Joerg Roedel te sale otra línea que pone que la funcionalidad no está disponible, no tienes soporte ni para IOMMUv1 ni para IOMMUv2. Ello teniendo en cuenta que no te has olvidado de activarlo en la BIOS, claro.

Si por el contrario la salida es algo como esto:

```
[ 1.637915] AMD-Vi: IOMMU performance counters supported
[ 1.640532] AMD-Vi: Found IOMMU at 0000:00:00.2 cap 0x40
[ 53.635666] AMD IOMMUv2 driver by Joerg Roedel <jroedel@suse.de>
```

Es decir, que se hace referencia a «Found IOMMU at...» y no hay una línea que diga que la funcionalidad de IOMMUv2 no está disponible, significa que tienes IOMMUv1 e IOMMUv2 soportado.

VUELVO A PONER LA NOTA: IOMMUv2 fue diseñado principalmente para las APUs con gráficos integrados como parte del HSA de AMD. A menos que estés tratando de pasar una tarjeta gráfica de una APU AMD (GPGPU) no necesitas IOMMUv2.

Si cumples con el requisito 2 pasa al

REQUISITO 3: SOPORTE PARA INTERRUPT REMAPPING

El hardware del host tiene que soportar interrupt remapping. No es posible utilizar PCI passthrough sin interrupt remapping.

Sistemas que NO soportan interrupt remapping

- Todos los sistemas con chipset y procesador AMD que tengan soporte para la tecnología de virtualización AMD I/O Virtualization (AMD-Vi) support. Si bien ese hardware tiene soporte para interrupt remapping, no hay software preparado para controlarla, al menos por ahora.
- Todos los sistemas con chipset y procesador Intel que tengan soporte para la tecnología Intel Virtualization Technology for Directed I/O (VT-d) pero que no tengan soporte para interrupt remapping.

Como saber si mi host ProxmoxVE tiene soporte para Interrupt remapping

Normalmente, si el hardware del host (chipset y procesador) donde tienes instalado ProxmoxVE es decentemente nuevo, seguramente contará con el soporte para esa tecnología y lo pondrá de forma clara. Si aún así no estás seguro hay una forma de saberlo. Para ello ejecuta:

```
dmesg | grep emapp
```

o

```
journalctl -k | grep emapp
```

Si el resultado es que tienes soporte para Interrupt Remaping puedes pasar al requisito 5. Si por el contrario el resultado es que no lo tienes, todavía tienes la opción de usar esa tecnología permitiendo las «interrupciones inseguras». Para ello ejecuta como root:

```
touch /etc/modprobe.d/pci-passthrough.conf
echo "options vfio_iommu_type1 allow_unsafe_interrupts=1" >> /etc/modprobe.d/pci-passthrough.conf
```

Una vez ejecutado ya puedes pasar al

REQUISITO 5: AGRUPAMIENTO ÚNICO DE IOMMU

Para que PCI passthrough funcione correctamente es necesario que la tarjeta PCI o PCIe a pasar a la máquina virtual esté dentro de su propio grupo IOMMU de manera que no comparta grupo con ningún otro dispositivo. Para comprobar que agrupamiento IOMMU tiene tu sistema, primero ejecuta lo siguiente como root:

Para procesadores Intel:

```
sed -i -e 's|GRUB_CMDLINE_LINUX_DEFAULT="quiet"|GRUB_CMDLINE_LINUX_DEFAULT="quiet intel_iommu=on"|g'
/etc/default/grub
update-grub
```

Para procesadores AMD:

```
sed -i -e 's|GRUB_CMDLINE_LINUX_DEFAULT="quiet"|GRUB_CMDLINE_LINUX_DEFAULT="quiet amd_iommu=on"|g'
/etc/default/grub
update-grub
```

Puedes agregar **iommu=pt** a la línea de ejecución del kernel para activar el etiquetado IOMMU sólo cuando sea necesario. Dará mejor rendimiento a los dispositivos configurados para ser pasados a máquinas virtuales. Si vas a pasar tarjetas gráficas, es aconsejado hacerlo de esa forma. La línea, si quieres usar iommu=pt te quedará de esta forma:

Para procesadores Intel:

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet intel_iommu=on iommu=pt"
```

Para procesadores AMD:

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet amd_iommu=on iommu=pt"
```

Una vez ejecutados los comandos anteriores se habrá agregado al archivo /etc/default/grub la opción para activar IOMMU y se habrá actualizado Grub con los cambios realizados. Entonces reinicia el ordenador con:

```
shutdown -r now
```

…y una vez reiniciado ya puedes crear el script para ver el agrupamiento IOMMU ejecutando como root:

```
echo '#!/bin/bash'                                                                          >
/root/MostrarGruposIOMMU.sh
echo ""                                                                                    >>
/root/MostrarGruposIOMMU.sh
echo 'echo ""'                                                                             >>
/root/MostrarGruposIOMMU.sh
echo 'echo "-------------------------------------------"'                                  >>
/root/MostrarGruposIOMMU.sh
echo 'echo " Grupos IOMMU disponibles en el sistema:"'                                     >>
/root/MostrarGruposIOMMU.sh
echo 'echo "-------------------------------------------"'                                  >>
/root/MostrarGruposIOMMU.sh
echo 'echo ""'                                                                             >>
/root/MostrarGruposIOMMU.sh
echo ""                                                                                    >>
/root/MostrarGruposIOMMU.sh
echo "for iommu_group in \$(find /sys/kernel/iommu_groups/ -maxdepth 1 -mindepth 1 -type d);" >>
/root/MostrarGruposIOMMU.sh
echo '  do echo "" && echo "Grupo IOMMU $(basename "$iommu_group")";'                     >>
/root/MostrarGruposIOMMU.sh
echo ""                                                                                    >>
/root/MostrarGruposIOMMU.sh
echo 'for device in $(ls -1 "$iommu_group"/devices/);'                                     >>
/root/MostrarGruposIOMMU.sh
echo '  do echo -n $'\t'; lspci -nns "$device"; done; done'                                >>
/root/MostrarGruposIOMMU.sh
echo ""                                                                                    >>
/root/MostrarGruposIOMMU.sh
echo 'echo ""'                                                                             >>
/root/MostrarGruposIOMMU.sh
chmod +x /root/MostrarGruposIOMMU.sh
```

… para luego correrlo ejecutando también como root:

```
/root/MostrarGruposIOMMU.sh
```

Si después de agregar **intel_iommu=on** o **amd_iommu=on** al archivo **/etc/default/grub**, de actualizar el grub, de reiniciar el sistema, de crear el script anterior y de ejecutarlo, el resultado de su ejecución es una salida nula, es decir, no se muestran grupos IOMMU entonces probablemente tu procesador no cuente con la característica ACS (Access Control Services). Todos los procesadores Xeon E3 y E5 tienen soporte para esa característica (excepto la familia E3-12xx). Con respecto a los Core de Intel, los que cuentan con ella, de los más usados, son:

Sandy Bridge-E (LGA2011)
i7-3960X (6-core, 3.3/3.9GHz)
i7-3970X (6-core, 3.5/4GHz)
i7-3930K (6-core, 3.2/3.8GHz)
i7-3820 (4-core, 3.6/3.8GHz)

Ivy Bridge-E (LGA2011)
i7-4960X (6-core, 3.6/4GHz)
i7-4930K (6-core, 3.4/3.6GHz)
i7-4820K (4-core, 3.7/3.9GHz)

Haswell-E (LGA2011-v3)
i7-5960X (8-core, 3/3.5GHz)
i7-5930K (6-core, 3.2/3.8GHz)
i7-5820K (6-core, 3.3/3.6GHz)

Broadwell-E (LGA2011-v3)
i7-6950X (10-core, 3.0/3.5GHz)
i7-6900K (8-core, 3.2/3.7GHz)
i7-6850K (6-core, 3.6/3.8GHz)
i7-6800K (6-core, 3.4/3.6GHz)

La mayoría de los i7 más nuevos que Haswell también deberían contar también con soporte ACS.

Si por otro lado, el resultado de la ejecución del script es algo como esto:

```
Grupo IOMMU 7
00:12.0 USB controller [0c03]: Advanced Micro Devices, Inc. [AMD] FCH USB EHCI Controller [1022:7908] (rev 49)

Grupo IOMMU 5
00:10.0 USB controller [0c03]: Advanced Micro Devices, Inc. [AMD] FCH USB XHCI Controller [1022:7914] (rev 20)

Grupo IOMMU 3
00:08.0 Encryption controller [1080]: Advanced Micro Devices, Inc. [AMD] Device [1022:1578]

Grupo IOMMU 1
00:02.0 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Device [1022:157b]
00:02.2 PCI bridge [0604]: Advanced Micro Devices, Inc. [AMD] Device [1022:157c]
00:02.5 PCI bridge [0604]: Advanced Micro Devices, Inc. [AMD] Device [1022:157c]
01:00.0 SATA controller [0106]: Marvell Technology Group Ltd. 88SE9230 PCIe SATA 6Gb/s Controller [1b4b:9230] (rev
11)
02:00.0 Ethernet controller [0200]: Broadcom Limited NetXtreme BCM5720 Gigabit Ethernet PCIe [14e4:165f]
02:00.1 Ethernet controller [0200]: Broadcom Limited NetXtreme BCM5720 Gigabit Ethernet PCIe [14e4:165f]

Grupo IOMMU 8
00:14.0 SMBus [0c05]: Advanced Micro Devices, Inc. [AMD] FCH SMBus Controller [1022:790b] (rev 4a)
00:14.3 ISA bridge [0601]: Advanced Micro Devices, Inc. [AMD] FCH LPC Bridge [1022:790e] (rev 11)

Grupo IOMMU 6
00:11.0 SATA controller [0106]: Advanced Micro Devices, Inc. [AMD] FCH SATA Controller [AHCI mode] [1022:7901] (rev
49)

Grupo IOMMU 4
00:09.0 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Device [1022:157d]

Grupo IOMMU 2
00:03.0 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Device [1022:157b]

Grupo IOMMU 0
00:01.0 VGA compatible controller [0300]: Advanced Micro Devices, Inc. [AMD/ATI] Carrizo [1002:9874] (rev 87)
00:01.1 Audio device [0403]: Advanced Micro Devices, Inc. [AMD/ATI] Kabini HDMI/DP Audio [1002:9840]

Grupo IOMMU 9
00:18.0 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Device [1022:1570]
00:18.1 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Device [1022:1571]
00:18.2 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Device [1022:1572]
00:18.3 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Device [1022:1573]
00:18.4 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Device [1022:1574]
00:18.5 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Device [1022:1575]
```

... entonces tu sistema soporta el agrupamiento IOMMU. Sólo te quedaría ver si la tarjeta que quieres pasar está aislada en su propio grupo. De ser así hay 99% de probabilidades de que PCI PassThrough sea exitoso. Si la tarjeta PCI o PCIe no está aislada en su propio grupo puedes intentar mover la tarjeta PCI y ver si ha quedado aislada o no.

Si después de moverla a otro slot tampoco has conseguido aislarla existe otra forma: el parche ACS de Alex Williamson. Pero **en ProxmoxVE no hace falta aplicar el parche** y recompilar el kernel dado que todos los kernels oficiales de ProxmoxVE ya vienen compilados con ese parche aplicado. Lo único que hay que hacer es activar la utilización de dicho parche mediante la edición del archivo:

```
/etc/default/grub
```

... donde agregaremos

```
pcie_acs_override=downstream
```

... a las opciones de la línea de comandos. Es decir, quedaría así:

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet intel_iommu=on pcie_acs_override=downstream"
```

Luego, actualizaríamos el grub con:

```
update-grub
```

... y después de reiniciar el sistema con:

```
shutdown -r now
```

... volver a ejecutar el script MostrarGruposIOMMU y ver si ese cambio dio resultado y la tarjeta quedó aislada o no.

Habitualmente, agregar esa última opción al grub ayuda a mejorar la mala implementación de ACS y a conseguir un mejor agrupamiento. Si tienes un Xeon E5, seguramente no te hará falta activar el parche porque los Xeon E5 consiguen un mejor agrupamiento que los E3 u otros procesadores en general.

Lo más probable es que, después de agregar el parche a la línea de comandos del kernel, una tarjeta gráfica pcie que quieras pasar a una máquina virtual, quede completamente aislada en su propio grupo. Pero si, además de pasar una tarjeta gráfica quieres pasar otras múltiples tarjetas PCIe y éstas no hayan quedado en su propio grupo, puedes empujar más la separación de los grupos IOMMU indicando al parche que sea «multifunción». Esto se hace editando la misma línea de antes y dejandola así:

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet intel_iommu=on pcie_acs_override=downstream,multifunction"
```

Lógicamente, después de esa modificación y al igual que las veces anteriores, tendrás que actualizar grub y reiniciar el sistema para ver los cambios.

Entonces, una vez logrado que la tarjeta a pasar esté dentro de su propio grupo, puedes proceder con la

CARGA DE LOS MÓDULOS VFIO AL INICIO DEL SISTEMA

Estos módulos buscan dispositivos PCI o PCIe que le hayas indicado mediante su PCI id en /etc/default/grub o en algún archivo de configuración dentro de /etc/modprobe.d/ y los «capturan» antes que cualquier otro módulo pueda intentar cargarlos y controlarlos. De esta forma esos dispositivos PCI indicados no son utilizados por el host y quedan libres para pasar a la máquina virtual sin tener que recurrir a «blacklistear» los módulos oficiales que los cargan. Todo esto es útil sobre todo para hacer PCI Passthrough de tarjetas gráficas, como comentaré en otro minitutorial. Entonces, vfio pone el dispositivo PCI a la «espera» hasta que otro programa pregunte por él. En este caso, quién preguntará por el dispositivo será el KVM de ProxmoxVE, y vfio se lo entregará. Pero tanto tengas pasado anular la carga de los módulos tradicionales con vfio como no anular nada, debes ejecutar las líneas de abajo si o si, porque los módulos vfio son necesarios para

hacer paso de dispositivos PCI o PCIe a la máquina virtual. Así que, para activar estos módulos, y contando con que ya agregaste IOMMU al grub, ejecuta como root:

```
echo "# PCIPassThrough" >> /etc/modules
echo "vfio"             >> /etc/modules
echo "vfio_iommu_type1" >> /etc/modules
echo "vfio_pci"         >> /etc/modules
echo "vfio_virqfd"      >> /etc/modules
```

... y reinicia el sistema.

Al finalizar de reiniciar, los módulos vfio deberían estar cargados. Lo siguiente es

CONFIGURAR EL PCI PASSTHROUGH EN LA MV

Una vez reiniciado el sistema puedes asignarle la tarjeta PCI a la máquina virtual localizándola primero con:

```
lspci
```

Ejecutado lo anterior te saldrá una lista como esta:

```
00:00.0 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Family 15h (Models 60h-6fh) Processor Root Complex
[1022:1576]
00:00.2 IOMMU [0806]: Advanced Micro Devices, Inc. [AMD] Family 15h (Models 60h-6fh) I/O Memory Management Unit
[1022:1577]
00:01.0 VGA compatible controller [0300]: Advanced Micro Devices, Inc. [AMD/ATI] Wani [Radeon R5/R6/R7 Graphics]
[1002:9874] (rev 84)
00:01.1 Audio device [0403]: Advanced Micro Devices, Inc. [AMD/ATI] Kabini HDMI/DP Audio [1002:9840]
00:02.0 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Family 15h (Models 60h-6fh) Host Bridge [1022:157b]
00:02.2 PCI bridge [0604]: Advanced Micro Devices, Inc. [AMD] Family 15h (Models 60h-6fh) Processor Root Port
[1022:157c]
00:02.4 PCI bridge [0604]: Advanced Micro Devices, Inc. [AMD] Family 15h (Models 60h-6fh) Processor Root Port
[1022:157c]
00:02.5 PCI bridge [0604]: Advanced Micro Devices, Inc. [AMD] Family 15h (Models 60h-6fh) Processor Root Port
[1022:157c]
00:03.0 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Family 15h (Models 60h-6fh) Host Bridge [1022:157b]
00:03.1 PCI bridge [0604]: Advanced Micro Devices, Inc. [AMD] Family 15h (Models 60h-6fh) Processor Root Port
[1022:157c]
00:08.0 Encryption controller [1080]: Advanced Micro Devices, Inc. [AMD] Carrizo Platform Security Processor
[1022:1578]
00:09.0 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Carrizo Audio Dummy Host Bridge [1022:157d]
00:10.0 USB controller [0c03]: Advanced Micro Devices, Inc. [AMD] FCH USB XHCI Controller [1022:7914] (rev 20)
00:11.0 SATA controller [0106]: Advanced Micro Devices, Inc. [AMD] FCH SATA Controller [AHCI mode] [1022:7901] (rev
49)
00:12.0 USB controller [0c03]: Advanced Micro Devices, Inc. [AMD] FCH USB EHCI Controller [1022:7908] (rev 49)
00:14.0 SMBus [0c05]: Advanced Micro Devices, Inc. [AMD] FCH SMBus Controller [1022:790b] (rev 4a)
00:14.3 ISA bridge [0601]: Advanced Micro Devices, Inc. [AMD] FCH LPC Bridge [1022:790e] (rev 11)
00:18.0 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Family 15h (Models 60h-6fh) Processor Function 0
[1022:1570]
00:18.1 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Family 15h (Models 60h-6fh) Processor Function 1
[1022:1571]
00:18.2 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Family 15h (Models 60h-6fh) Processor Function 2
[1022:1572]
00:18.3 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Family 15h (Models 60h-6fh) Processor Function 3
[1022:1573]
00:18.4 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Family 15h (Models 60h-6fh) Processor Function 4
[1022:1574]
00:18.5 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Family 15h (Models 60h-6fh) Processor Function 5
[1022:1575]
01:00.0 SATA controller [0106]: Marvell Technology Group Ltd. 88SE9230 PCIe 2.0 x2 4-port SATA 6 Gb/s RAID
Controller [1b4b:9230] (rev 11)
02:00.0 PCI bridge [0604]: PLX Technology, Inc. PEX 8606 6 Lane, 6 Port PCI Express Gen 2 (5.0 GT/s) Switch
```

```
[10b5:8606] (rev ba)
03:01.0 PCI bridge [0604]: PLX Technology, Inc. PEX 8606 6 Lane, 6 Port PCI Express Gen 2 (5.0 GT/s) Switch
[10b5:8606] (rev ba)
03:04.0 PCI bridge [0604]: PLX Technology, Inc. PEX 8606 6 Lane, 6 Port PCI Express Gen 2 (5.0 GT/s) Switch
[10b5:8606] (rev ba)
03:05.0 PCI bridge [0604]: PLX Technology, Inc. PEX 8606 6 Lane, 6 Port PCI Express Gen 2 (5.0 GT/s) Switch
[10b5:8606] (rev ba)
03:07.0 PCI bridge [0604]: PLX Technology, Inc. PEX 8606 6 Lane, 6 Port PCI Express Gen 2 (5.0 GT/s) Switch
[10b5:8606] (rev ba)
03:09.0 PCI bridge [0604]: PLX Technology, Inc. PEX 8606 6 Lane, 6 Port PCI Express Gen 2 (5.0 GT/s) Switch
[10b5:8606] (rev ba)
06:00.0 Network controller [0280]: Qualcomm Atheros QCA9984 802.11ac Wave 2 Wireless Network Adapter [168c:0046]
(rev 01)
07:00.0 Network controller [0280]: Qualcomm Atheros QCA9984 802.11ac Wave 2 Wireless Network Adapter [168c:0046]
(rev 01)
09:00.0 Ethernet controller [0200]: Broadcom Inc. and subsidiaries NetXtreme BCM5720 2-port Gigabit Ethernet PCIe
[14e4:165f]
09:00.1 Ethernet controller [0200]: Broadcom Inc. and subsidiaries NetXtreme BCM5720 2-port Gigabit Ethernet PCIe
[14e4:165f]
0a:00.0 PCI bridge [0604]: Microsemi / PMC / IDT PES12N3A 12-lane 3-Port PCI Express Switch [111d:8018] (rev 0e)
0b:02.0 PCI bridge [0604]: Microsemi / PMC / IDT PES12N3A 12-lane 3-Port PCI Express Switch [111d:8018] (rev 0e)
0b:04.0 PCI bridge [0604]: Microsemi / PMC / IDT PES12N3A 12-lane 3-Port PCI Express Switch [111d:8018] (rev 0e)
0c:00.0 Ethernet controller [0200]: Intel Corporation 82575GB Gigabit Network Connection [8086:10d6] (rev 02)
0c:00.1 Ethernet controller [0200]: Intel Corporation 82575GB Gigabit Network Connection [8086:10d6] (rev 02)
0d:00.0 Ethernet controller [0200]: Intel Corporation 82575GB Gigabit Network Connection [8086:10d6] (rev 02)
0d:00.1 Ethernet controller [0200]: Intel Corporation 82575GB Gigabit Network Connection [8086:10d6] (rev 02)
```

En mi caso la tarjeta a pasar es:

```
01:00.0 SATA controller [0106]: Marvell Technology Group Ltd. 88SE9230 PCIe 2.0 x2 4-port SATA 6 Gb/s RAID
Controller [1b4b:9230] (rev 11)
```

Por lo que debería tomar nota de los números que aparecen al principio de la línea (**01:00.0)** y de su pciid (**1b4b:9230**).

AGREGAR A pci-passthrough.conf EL PCIID DE LA TARJETA A PASAR

Agrega al archivo /etc/modprobe.d/pci-passthrough.conf el PCIID de la tarjeta/s que quieras pasar de la siguiente forma:

```
options vfio-pci ids=1b4b:9230
```

Después de editar el archivo /etc/modprobe.d/pci-passthrough.conf (o cualquier archivo dentro de /etc/modprobe.d/) siempre debemos ejecutar:

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

PASAR LA TARJETA SI ES PCI

Habiendo tomado nota del número anterior, para pasar la tarjeta a la vm debería editar manualmente su archivo de configuración, según sea su ID con, por ejemplo:

```
nano /etc/pve/qemu-server/100.conf
```

y asignarle la tarjeta agregando a ese archivo la línea:

```
hostpci0: 01:00.0
```

…para luego guardarlo e iniciar la máquina virtual.

Si la tarjeta a asignar tiene más de un dispositivo dentro, por ejemplo

01:00.0 y 01:00.1, puedes asignarlos manualmente a todos agregando la línea:

```
hostpci0: 01:00.0;01:00.1
```

O sino puedes asignar de forma automática todos los dispositivos de dentro de una tarjeta agregando la línea:

```
hostpci0: 01:00
```

PASAR LA TARJETA SI ES PCI EXPRESS

Si la tarjeta es PCIe y no PCI, para pasarla tienes que agregar al archivo de configuración algunos datos extra. Por ejemplo, para pasar una tarjeta 01:00.0 tendrías que agregar al archivo de configuración las líneas:

```
machine: q35
hostpci0: 01:00.0,pcie=1
```

Saca tus conclusiones que para eso eres un geek.