Pasar tarjetas de expansión a una máquina virtual.

Mario Laglera Barco

Índice

1. Intruducción.

En un host de máquinas virtuales es posible darle acceso a una máquina virtual de una tarjeta de expansión como una controladora de discos o una controladora de ethernet.

Gracias a esto podemos crear una máquina virtual con TrueNAS de sistema operativo, porque en los sistemas NAS es recomendable que el sistema acceda directamente a los discos.

También podemos tener una máquina virtual con acceso a una tarjeta gráfica y ejecutar aplicaciones que se vean beneficiadas de esto, como Adobe Premiere Pro, DaVinci Resolve o Blender, por decir unos ejemplos.

Y un último ejemplo es la creación de un router con el sistema operativo PFsense.

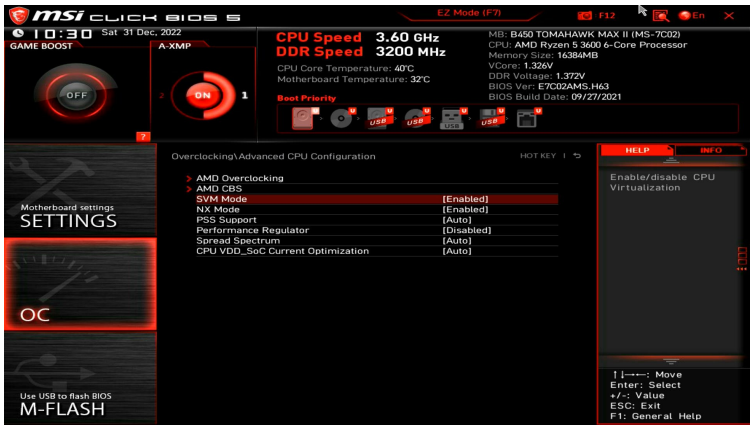
Todo esto lo podemos hacer es cualquier distribución Linux, aunque lo más remcomendable es instalar algo más preparado como Proxmox.

2. Configuración del host.

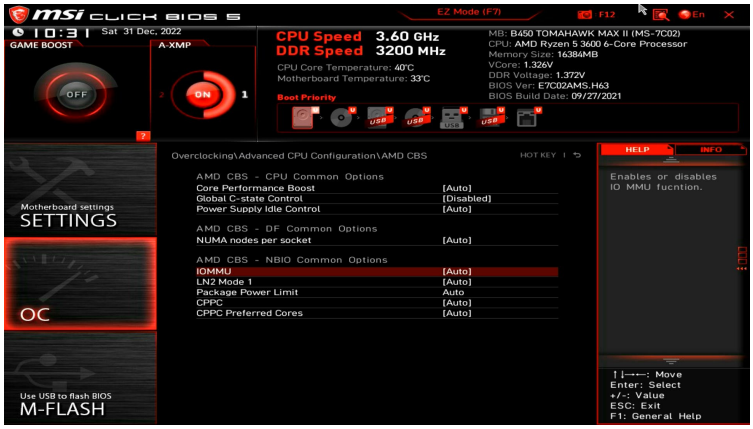
En el equipo host tenemos que cambiar unas cosas dentro del BIOS, instalar el sistema operativo y configurar este sistema operativo.

2.1. Configuración del BIOS.

Las configuraciones que tenemos que activar dentro del BIOS son la primera la virtualización, para que la máquina virtual pueda ser acelerada por hardware.



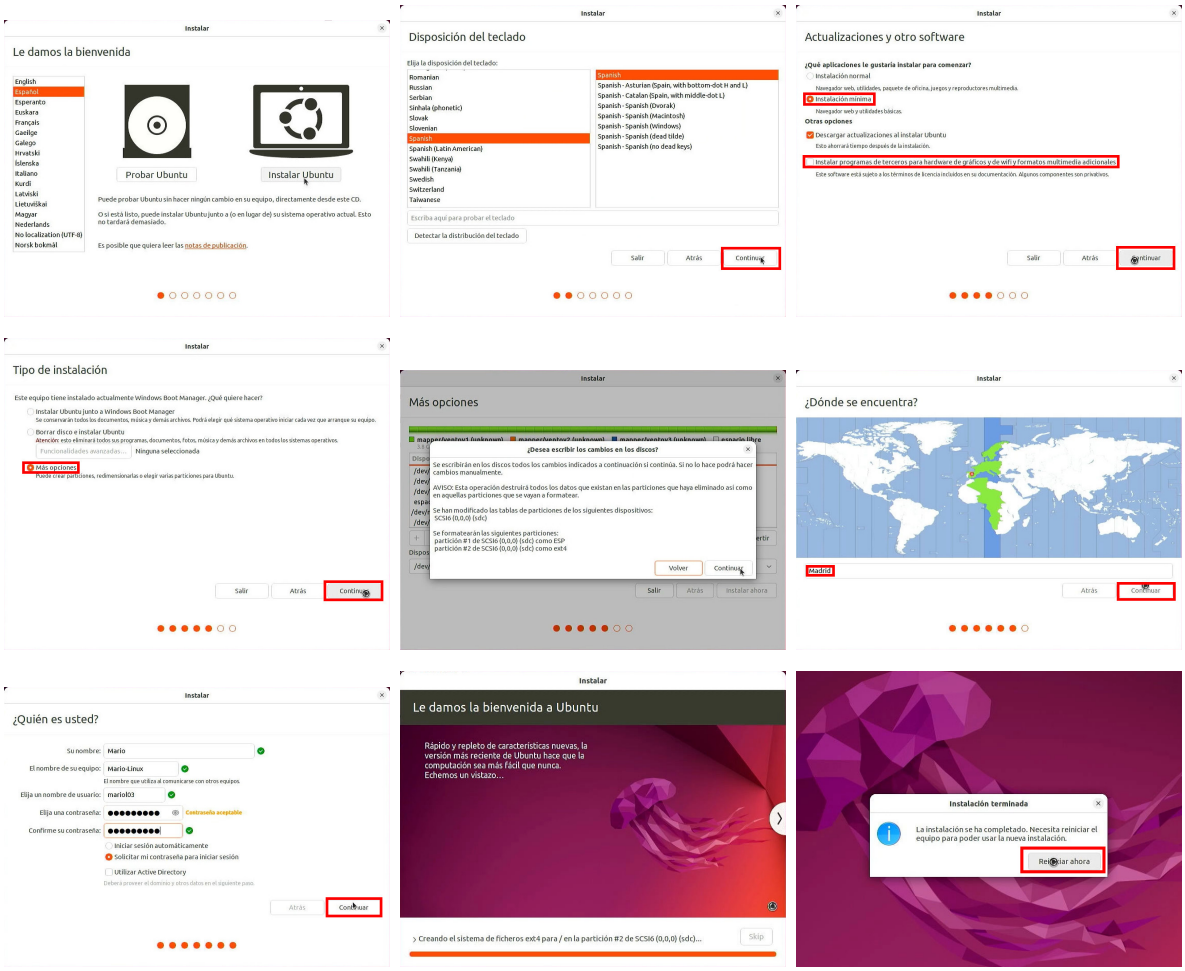
Y la otra opción que debemos cambiar es activar el gestor de memoria de entrada y salida (IOMMU).



También dependiendo del hardware tendremos que activar el ReBAR (permitir que el procesador acceda a toda la memoria de una tarjeta de expansión) y otra cosa que se puede activar es el SR-IOV (que en tarjetas de expansión que lo soportadas, permite compartir una parte de ella a una máquina virtual y otra parte a otra máquina virtual).

2.2. Instalación del sistema operativo.

El sistema opertaivo que selecione fue Ubuntu Desktop 22.04, en el momento de la instalación no se instaló los drivers propietarios de Nvidia y en una unidad de estado sólido separado.



2.3. Preparación del sistema operativo.

2.3.1. Comporbaciones previas.

Una vez instalado el sistema tenemos que comporbar los grupos de memoria estan activados y que la tarjeta de expansión que queremos pasar a la máquina virtual esté en uno propio, para ello se ejecuta en una terminal *dmesg | grep "IOMMU"* y nos mostrará la id del dispositivo y

el número del grupo de memoria. Tambien podemos ejecutar un script como este para que lo muestre de una manera más sencilla.

```
#!/bin/bash
for d in $(find /sys/kernel/iommu_groups/ -type l \ sort -n -k5 -t/); do
  n=${d#*/iommu_groups/*}; n=${n%%/*}
  printf 'IOMMU Group %s ' "$n"
  lspci -nns "${d##*/}"
done;

```

Que nos mostrará algo similar a esto.

```
IOMMU Group 0 00:01.0 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Starship/Matisse PCIe Dummy Host Bridge [1022:1482]
IOMMU Group 1 00:01.1 PCI bridge [0604]: Advanced Micro Devices, Inc. [AMD] Starship/Matisse GPP Bridge [1022:1483]
IOMMU Group 2 00:01.3 PCI bridge [0604]: Advanced Micro Devices, Inc. [AMD] Starship/Matisse GPP Bridge [1022:1483]
IOMMU Group 3 00:02.0 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Starship/Matisse PCIe Dummy Host Bridge [1022:1482]
IOMMU Group 4 00:03.0 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Starship/Matisse PCIe Dummy Host Bridge [1022:1482]
IOMMU Group 5 00:03.1 PCI bridge [0604]: Advanced Micro Devices, Inc. [AMD] Starship/Matisse GPP Bridge [1022:1483]
IOMMU Group 6 00:04.0 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Starship/Matisse PCIe Dummy Host Bridge [1022:1482]
IOMMU Group 7 00:05.0 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Starship/Matisse PCIe Dummy Host Bridge [1022:1482]
IOMMU Group 8 00:07.0 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Starship/Matisse PCIe Dummy Host Bridge [1022:1482]
IOMMU Group 9 00:07.1 PCI bridge [0604]: Advanced Micro Devices, Inc. [AMD] Starship/Matisse Internal PCIe GPP Bridge 0 to bus[E:B] [1022:1484]
IOMMU Group 10 00:08.0 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Starship/Matisse PCIe Dummy Host Bridge [1022:1482]
IOMMU Group 11 00:08.1 PCI bridge [0604]: Advanced Micro Devices, Inc. [AMD] Starship/Matisse Internal PCIe GPP Bridge 0 to bus[E:B] [1022:1484]
IOMMU Group 12 00:14.0 SMBus [0c05]: Advanced Micro Devices, Inc. [AMD] FCH SMBus Controller [1022:790b] (rev 61)
IOMMU Group 12 00:14.3 ISA bridge [0601]: Advanced Micro Devices, Inc. [AMD] FCH LPC Bridge [1022:790e] (rev 51)
IOMMU Group 13 00:18.0 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Matisse/Vermeer Data Fabric: Device 18h; Function 0 [1022:1440]
IOMMU Group 13 00:18.1 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Matisse/Vermeer Data Fabric: Device 18h; Function 1 [1022:1441]
IOMMU Group 13 00:18.2 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Matisse/Vermeer Data Fabric: Device 18h; Function 2 [1022:1442]
IOMMU Group 13 00:18.3 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Matisse/Vermeer Data Fabric: Device 18h; Function 3 [1022:1443]
IOMMU Group 13 00:18.4 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Matisse/Vermeer Data Fabric: Device 18h; Function 4 [1022:1444]
IOMMU Group 13 00:18.5 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Matisse/Vermeer Data Fabric: Device 18h; Function 5 [1022:1445]
IOMMU Group 13 00:18.6 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Matisse/Vermeer Data Fabric: Device 18h; Function 6 [1022:1446]
IOMMU Group 13 00:18.7 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Matisse/Vermeer Data Fabric: Device 18h; Function 7 [1022:1447]
IOMMU Group 14 01:00.0 Non-Volatile memory controller [0108]: Kingston Technology Company, Inc. Device [2646:5017] (rev 03)
IOMMU Group 15 03:00.0 USB controller [0c03]: Advanced Micro Devices, Inc. [AMD] 400 Series Chipset USB 3.1 XHCI Controller [1022:43d5] (rev 01)
IOMMU Group 15 03:00.1 SATA controller [0106]: Advanced Micro Devices, Inc. [AMD] 400 Series Chipset SATA Controller [1022:43c8] (rev 01)
IOMMU Group 15 03:00.2 PCI bridge [0604]: Advanced Micro Devices, Inc. [AMD] 400 Series Chipset PCIe Bridge [1022:43c6] (rev 01)
IOMMU Group 15 20:00.0 PCI bridge [0604]: Advanced Micro Devices, Inc. [AMD] 400 Series Chipset PCIe Port [1022:43c7] (rev 01)
IOMMU Group 15 20:01.0 PCI bridge [0604]: Advanced Micro Devices, Inc. [AMD] 400 Series Chipset PCIe Port [1022:43c7] (rev 01)
IOMMU Group 15 20:04.0 PCI bridge [0604]: Advanced Micro Devices, Inc. [AMD] 400 Series Chipset PCIe Port [1022:43c7] (rev 01)
IOMMU Group 15 21:00.0 Ethernet controller [0200]: Intel Corporation Ethernet Controller I225-V [8086:15f3] (rev 03)
IOMMU Group 15 22:00.0 Ethernet controller [0200]: Realtek Semiconductor Co., Ltd. RTL8111/8168/8411 PCI Express Gigabit Ethernet Controller [10ec:8168] (rev 15)
IOMMU Group 16 26:00.0 VGA compatible controller [0300]: NVIDIA Corporation TU117 [GeForce GTX 1650] [10de:1f82] (rev a1)
IOMMU Group 16 26:00.1 Audio device [0403]: NVIDIA Corporation Device [10de:10fa] (rev a1)
IOMMU Group 17 27:00.0 Non-Essential Instrumentation [1300]: Advanced Micro Devices, Inc. [AMD] Starship/Matisse PCIe Dummy Function [1022:148a]
IOMMU Group 18 28:00.0 Non-Essential Instrumentation [1300]: Advanced Micro Devices, Inc. [AMD] Starship/Matisse Reserved SPP [1022:1485]
IOMMU Group 19 28:00.1 Encryption controller [1080]: Advanced Micro Devices, Inc. [AMD] Starship/Matisse Cryptographic Coprocessor PSPCPP [1022:1486]
IOMMU Group 20 28:00.3 USB controller [0c03]: Advanced Micro Devices, Inc. [AMD] Matisse USB 3.0 Host Controller [1022:149c]
IOMMU Group 21 28:00.4 Audio device [0403]: Advanced Micro Devices, Inc. [AMD] Starship/Matisse HD Audio Controller [1022:1487]
```

2.3.2. Separar los grupos aún más.

Es posible que para el proposito que queremos no nos sirva la separación que tenemos, como está definido por hardware pordemos intentar arreglarlo moviendo la tarjeta a otro puerto, en mi caso no se solucionó para arreglar se puede aplicar un parche del kernel llamado acs, para ello podemos descargarnos un kernel compilado e instalarlo. Una vez lo hemos instalado tendremos que cambiar la configuracion del gestor de arranque para que arranque con este nuevo kernel y que aplique las configuraciones del parche, para ello modificamos las lineas GRUB_DEFAULT y GRUB_CMDLINE_LINUX_DEFAULT.

```
GRUB_DEFAULT="1>4"
GRUB_TIMEOUT_STYLE=hidden
GRUB_TIMEOUT=1
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash vfio-pci.ids=8086:15f3 pcie_acs_override=downstream,multifunction"
GRUB_CMDLINE_LINUX=""
```

Y luego ejecutamos el comando *update-grub* para que los cambios se queden guardados y reiniciamos. Una vez se halla reiniciado volvemos a ver los grupos IOMMU y cada dispositivo deberia de estar en un grupo separado.

```
IOMMU Group 0 00:01.0 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Starship/Matisse PCIe Dummy Host Bridge [1022:1482]
IOMMU Group 1 00:01.1 PCI bridge [0604]: Advanced Micro Devices, Inc. [AMD] Starship/Matisse GPP Bridge [1022:1483]
IOMMU Group 2 00:01.3 PCI bridge [0604]: Advanced Micro Devices, Inc. [AMD] Starship/Matisse GPP Bridge [1022:1483]
IOMMU Group 3 00:02.0 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Starship/Matisse PCIe Dummy Host Bridge [1022:1482]
IOMMU Group 4 00:03.0 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Starship/Matisse PCIe Dummy Host Bridge [1022:1482]
IOMMU Group 5 00:03.1 PCI bridge [0604]: Advanced Micro Devices, Inc. [AMD] Starship/Matisse GPP Bridge [1022:1483]
IOMMU Group 6 00:04.0 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Starship/Matisse PCIe Dummy Host Bridge [1022:1482]
IOMMU Group 7 00:05.0 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Starship/Matisse PCIe Dummy Host Bridge [1022:1482]
IOMMU Group 8 00:07.0 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Starship/Matisse PCIe Dummy Host Bridge [1022:1482]
IOMMU Group 9 00:07.1 PCI bridge [0604]: Advanced Micro Devices, Inc. [AMD] Starship/Matisse Internal PCIe GPP Bridge 0 to bus[E:B] [1022:1484]
IOMMU Group 10 00:08.0 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Starship/Matisse PCIe Dummy Host Bridge [1022:1482]
IOMMU Group 11 00:08.1 PCI bridge [0604]: Advanced Micro Devices, Inc. [AMD] Starship/Matisse Internal PCIe GPP Bridge 0 to bus[E:B] [1022:1484]
IOMMU Group 12 00:14.0 SMBus [0c05]: Advanced Micro Devices, Inc. [AMD] FCH SMBus Controller [1022:790b] (rev 61)
IOMMU Group 12 00:14.3 ISA bridge [0601]: Advanced Micro Devices, Inc. [AMD] FCH LPC Bridge [1022:790e] (rev 51)
IOMMU Group 13 00:18.0 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Matisse/Vermeer Data Fabric: Device 18h; Function 0 [1022:1440]
IOMMU Group 13 00:18.1 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Matisse/Vermeer Data Fabric: Device 18h; Function 1 [1022:1441]
IOMMU Group 13 00:18.2 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Matisse/Vermeer Data Fabric: Device 18h; Function 2 [1022:1442]
IOMMU Group 13 00:18.3 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Matisse/Vermeer Data Fabric: Device 18h; Function 3 [1022:1443]
IOMMU Group 13 00:18.4 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Matisse/Vermeer Data Fabric: Device 18h; Function 4 [1022:1444]
IOMMU Group 13 00:18.5 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Matisse/Vermeer Data Fabric: Device 18h; Function 5 [1022:1445]
IOMMU Group 13 00:18.6 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Matisse/Vermeer Data Fabric: Device 18h; Function 6 [1022:1446]
IOMMU Group 13 00:18.7 Host bridge [0600]: Advanced Micro Devices, Inc. [AMD] Matisse/Vermeer Data Fabric: Device 18h; Function 7 [1022:1447]
IOMMU Group 14 01:00.0 Non-Volatile memory controller [0108]: Kingston Technology Company, Inc. Device [2646:5017] (rev 03)
IOMMU Group 15 03:00.0 USB controller [0c03]: Advanced Micro Devices, Inc. [AMD] 400 Series Chipset USB 3.1 XHCI Controller [1022:43d5] (rev 01)
IOMMU Group 16 03:00.1 SATA controller [0106]: Advanced Micro Devices, Inc. [AMD] 400 Series Chipset SATA Controller [1022:43c8] (rev 01)
IOMMU Group 17 03:00.2 PCI bridge [0604]: Advanced Micro Devices, Inc. [AMD] 400 Series Chipset PCIe Bridge [1022:43c6] (rev 01)
IOMMU Group 18 20:00.0 PCI bridge [0604]: Advanced Micro Devices, Inc. [AMD] 400 Series Chipset PCIe Port [1022:43c7] (rev 01)
IOMMU Group 19 20:01.0 PCI bridge [0604]: Advanced Micro Devices, Inc. [AMD] 400 Series Chipset PCIe Port [1022:43c7] (rev 01)
IOMMU Group 20 20:04.0 PCI bridge [0604]: Advanced Micro Devices, Inc. [AMD] 400 Series Chipset PCIe Port [1022:43c7] (rev 01)
IOMMU Group 21 21:00.0 Ethernet controller [0200]: Intel Corporation Ethernet Controller I225-V [8086:15f3] (rev 03)
IOMMU Group 22 22:00.0 Ethernet controller [0200]: Realtek Semiconductor Co., Ltd. RTL8111/8168/8411 PCI Express Gigabit Ethernet Controller [10ec:8168] (rev 15)
IOMMU Group 23 26:00.0 VGA compatible controller [0300]: NVIDIA Corporation TU117 [GeForce GTX 1650] [10de:1f82] (rev a1)
IOMMU Group 24 26:00.1 Audio device [0403]: NVIDIA Corporation Device [10de:10fa] (rev a1)
IOMMU Group 25 27:00.0 Non-Essential Instrumentation [1300]: Advanced Micro Devices, Inc. [AMD] Starship/Matisse PCIe Dummy Function [1022:148a]
IOMMU Group 26 28:00.0 Non-Essential Instrumentation [1300]: Advanced Micro Devices, Inc. [AMD] Starship/Matisse Reserved SPP [1022:1485]
IOMMU Group 27 28:00.1 Encryption controller [1080]: Advanced Micro Devices, Inc. [AMD] Starship/Matisse Cryptographic Coprocessor PSPCPP [1022:1486]
IOMMU Group 28 28:00.3 USB controller [0c03]: Advanced Micro Devices, Inc. [AMD] Matisse USB 3.0 Host Controller [1022:149c]
IOMMU Group 29 28:00.4 Audio device [0403]: Advanced Micro Devices, Inc. [AMD] Starship/Matisse HD Audio Controller [1022:1487]
```

2.3.3. Instalacion del software necesario.

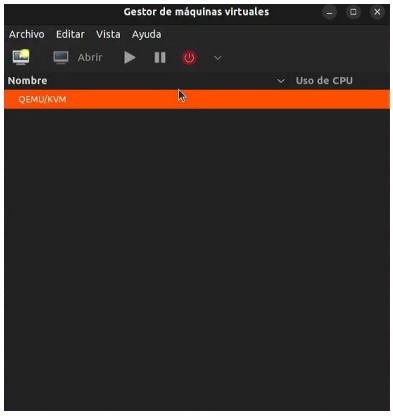
Una vez ya tenemos los grupos IOMMU separados podemos instalar el hipervisor y el gestor gráfico de este. Para ello abirmos una terminal y ejecutamos el comando *apt install virt-manager*

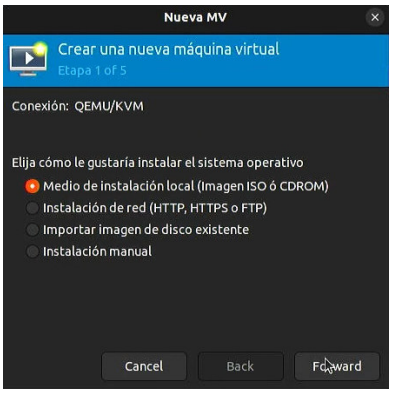
3. Creación y configuración de la máquina virtual.

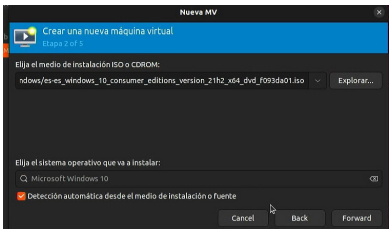
Una vez tengamos el host configurado y preparado podemos crear una maquina virtual, yo voy a crear una maquina Windows 10 para enseñar el uso de software accelerado por hardware (Tarjeta gráfica).

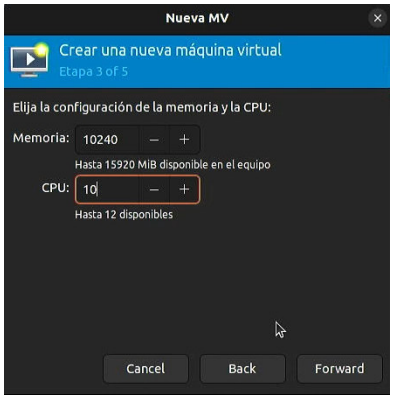
3.1. Creación de la máquina virtual.

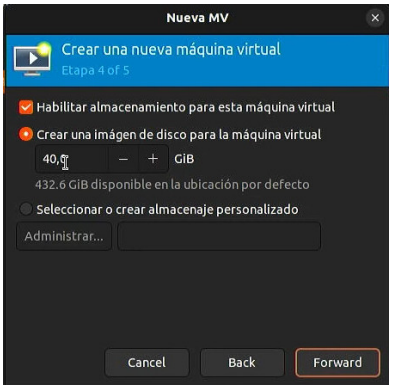
Para crear la maquina virtual tenemos que abrir el gestor de manquinas virtuales (virt-manager). Una vez dentro de esta aplicación nos saldra la conexion al hypervisor local, es posible que nos pida autenticarnos ya que por defecto no tenemos permisos para virtualizar, una vez estemos autenticados podremos pulsar el botón de nuevo e indicarle la iso de instalación, el sistema operativo que vamos a instalar y los recursos que tendra la máquina virtual.

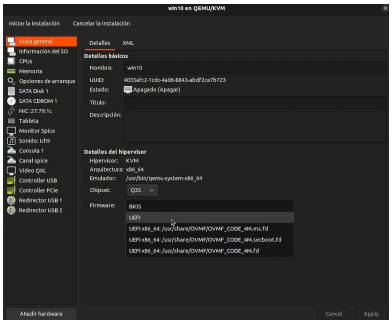




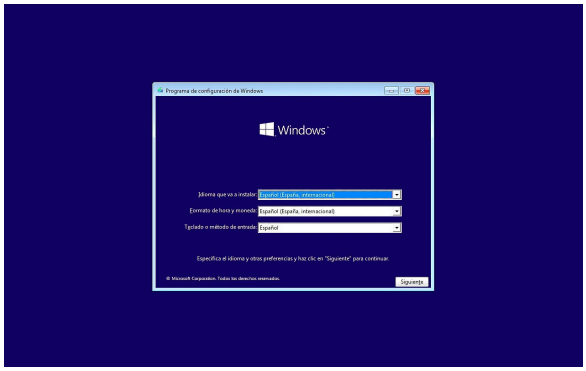


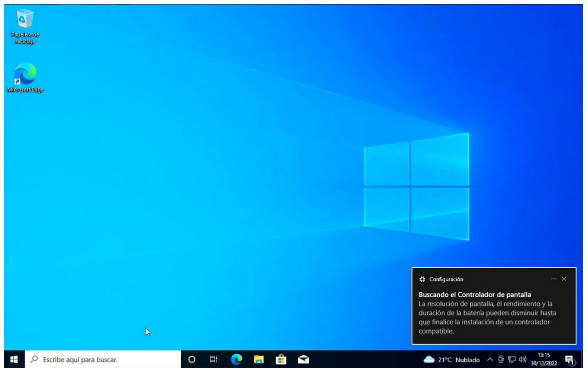






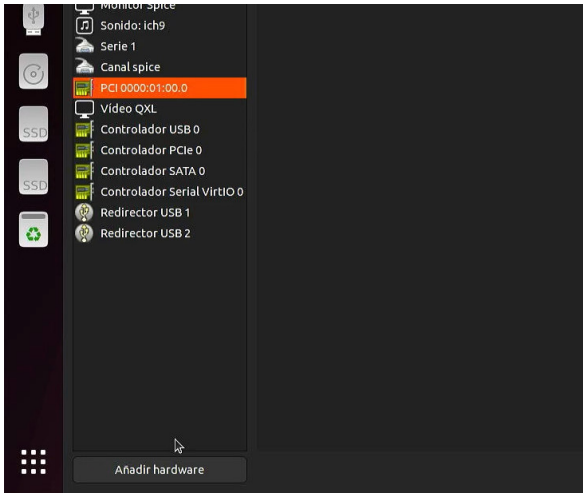
Una vez creada se abirará e instalamos Windows como en un equipo normal.

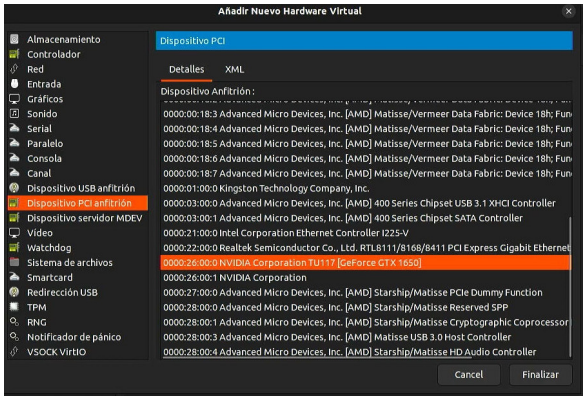




3.2. Modificar la máquina virtual para pasar la tarjeta de expansión.

Ahora que tenemos windows instalado abirmos la maquina virtual y sin encenderla vamos a la pestaña de configuración y dentro de añadir hardware seleccionamos "Dispositivo PCI anfitrión" y el dispositivo que queramos usar.





Para pasar una tarjeta de expansion, se tiene que pasar esta entera, por ejemplo en una tarjeta gráfica se tiene que pasar la GPU y el controlador de Audio (usado para las salidas HDMI).
En el caso de que solo tengamos una GPU tenemos que usar unos scripts para que decirle a linux que la deje de usar sin esto la maquina no enciende ya que el kernel de linux sigue usando los drivers. Para ello podemos usar los "ganchos" de libvirt para que ejecute un script antes del arranque de la maquina y descarge los drivers.
Si usamos una tarjeta de expansión que no sea una GPU si encendemos la MV funcionaria sin tener que desactivar nada, ya que con la configuracion previamente realizada ya teniamos los grupos IOMMU configurados.

3.2.1. Habilitar los scritps gancho.

Lo primero que tenemos que hacer es habilitar estos scripts para que se ejecuten en el momento que una maquina virtual se encienda o se apague, para ello tenemos que crear una carpeta llamada hooks dentro de /etc/libvirt/ con el comando *mkdir /etc/libvirt/hooks*. Una vez creada tenemos que descargar el ejecutable para que se ejecuten los scripts que podemos hacerlo con el comando *sudo wget 'https://raw.githubusercontent.com/PassthroughPOST/VFIO-Tools/master/libvirt_hooks/qemu' -O /etc/libvirt/hooks/qemu* y luego darle permisos de ejecución con el comando *sudo chmod +x /etc/libvirt/hooks/qemu*

3.2.2. Configuracion de los scripts gancho para nuestra MV.

Una vez habilitados tenemos que crear las carpetas de preparacion y de salida dentro de una con el nombre de nuestra maquina en la carpeta /etc/libvirt/hooks/qemu.d/ para ello podemos crear las ultimas carpetas con los comandos *sudo mkdir -p /etc/libvirt/hooks/qemu.d/win10/prepare/begin* y *sudo mkdir -p /etc/libvirt/hooks/qemu.d/win10/release/end.*

3.2.3. Creacion de los scripts.

Tenemos que crear dos scripts unos para que se ejecute en el momento de arranque y otro en el apagado de la maquina.
En el script de arranque tenemos que poner algo como esto, en el que tenemos que varir la id del dispositvivo PCIe y los drivers que usa nuestra gráfica.

```
#!/bin/bash
systemctl stop display-manager.service
echo 0 > /sys/class/vtconsole/vtcon0/bind
echo 0 > /sys/class/vtconsole/vtcon1/bind
echo efi-framebuffer.0 > /sys/bus/platform/drivers/efi-framebuffer/unbind
rmmod nvidia_uvm
rmmod nvidia_drm
rmmod nvidia_modeset
rmmod nvidia
sleep 2
virsh nodedev-detach pci_0000_26_00_0
virsh nodedev-detach pci_0000_26_00_1

```

Conjunto de comandos 1: Script de arranque

Este lo ponemos en la carpeta /etc/libvirt/hooks/qemu.d/win10/prepare/begin/ con el nombre start.sh

```
#!/bin/bash
virsh nodedev-reattach pci_0000_26_00_1
virsh nodedev-reattach pci_0000_26_00_0
modprobe nvidia
modprobe nvidia_modeset
modprobe nvidia_uvm
modprobe nvidia_drm
echo 1 > /sys/class/vtconsole/vtcon0/bind
echo 1 > /sys/class/vtconsole/vtcon1/bind
nvidia-xconfig --query-gpu-info > /dev/null 2>&1
echo "efi-framebuffer.0" > /sys/bus/platform/drivers/efi-framebuffer/bind
systemctl start display-manager.service

```

Conjunto de comandos 2: Script de apagado

Y este lo ponemos en la carpeta /etc/libvirt/hooks/qemu.d/win10/release/end/ con el nombre revert.sh

3.2.4. Añadir la VBIOS a la MV.

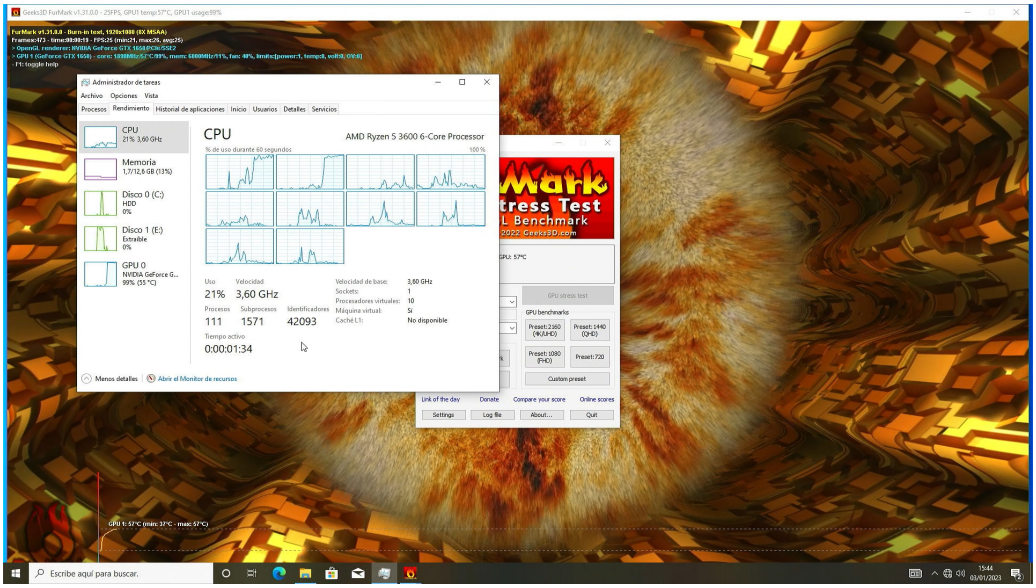
Aunque tengamos la GPU añadida a la MV y quizas se vea la BIOS de la MV cuando windows carge los drivers nos dara un error ya que estos drivers no son capaces de saber que grafica es correctamente al faltarle acceso a la ROM de la gráfica esto lo podemos arreglar adjuntando una copia, previamente hecha con GPU-Z o programas similares, a la MV para ello añadimos la siguiente linea a la configuracion de la grafica *<rom file='/var/lib/libvirt/images/TU117.rom' />*. Una vez hecho esto ya podemos encender la MV e instalar lo drivers como si fuera un equipo normal.

4. Conclusiones.

Concluyendo, es un proceso interesante para virtualizar las aplicaciones que requieran el acceso a esa tarjeta de expansión, aunque el rendimiento en la maquina real sera superior, ya que se tiene acceso a todo el hardware. También permite crear redes con una salida WAN totalmente virtualizar o virutalizar un router si se tiene dos o más tarjetas de red.

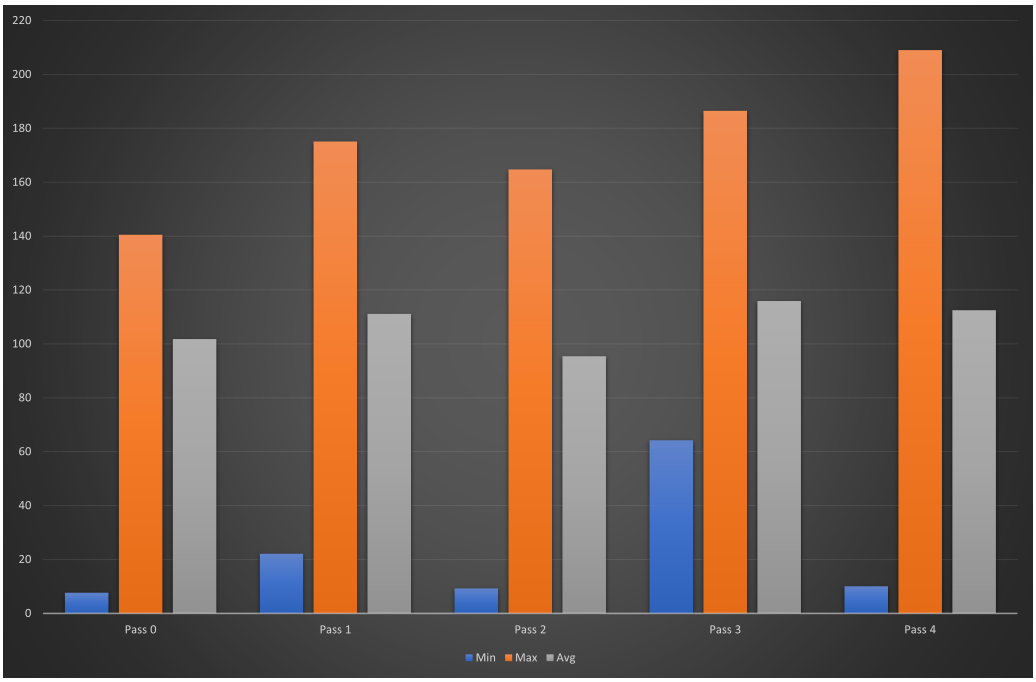
4.1. Rendimiento en MV y en baremetal en Furmark.

Rendimento en Bare Metal

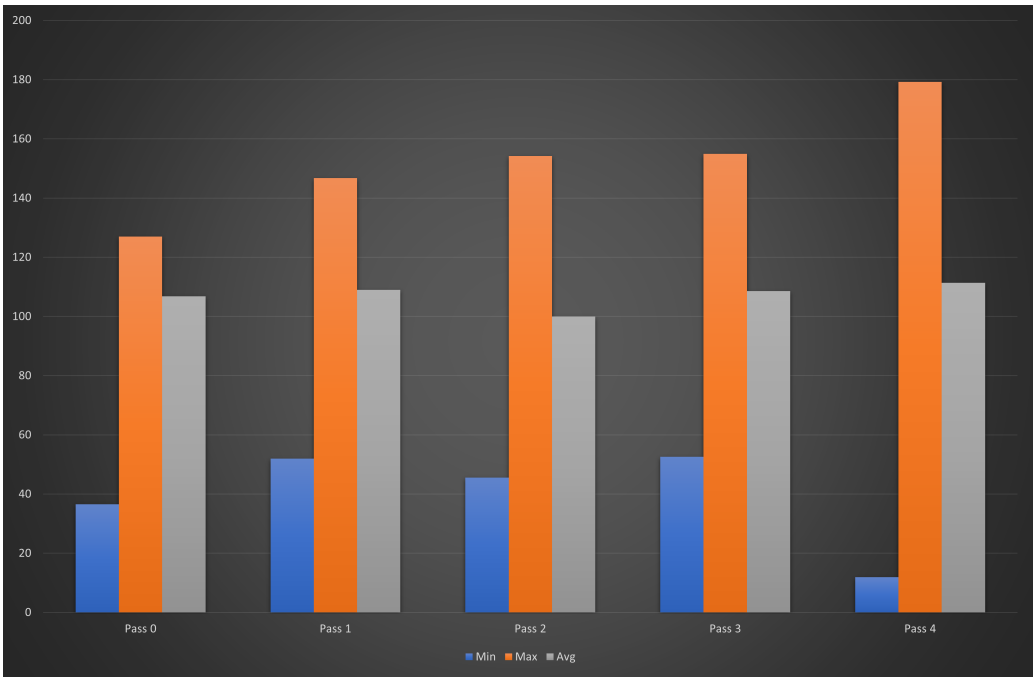


Rendimento en Máquina Virtual

4.2. Rendimiento en MV y baremetal en otras aplicaciones.



Rendimento en Bare Metal, video: youtube.com



Rendimento en Máquina Virtual, video: youtube.com