Cuando tenemos **varias tarjetas gráficas instaladas en un servidor Proxmox**, podemos encontrarnos con un problema particular si queremos utilizar una de ellas para el propio host y reservar otra completamente para una máquina virtual mediante **PCI Passthrough**.

Un escenario bastante habitual consiste en tener una tarjeta NVIDIA instalada en el primer puerto PCIe de la placa base y una AMD instalada en el segundo. Podemos querer que **Proxmox utilice exclusivamente la AMD**, mientras que la NVIDIA permanece vinculada a **vfio-pci** para poder entregarla directamente a una máquina virtual.

El problema aparece cuando tenemos monitores conectados a ambas tarjetas gráficas. Aunque tengamos los controladores de NVIDIA bloqueados y la tarjeta correctamente vinculada a vfio-pci, **la UEFI puede inicializar igualmente la NVIDIA como tarjeta gráfica principal** simplemente porque se encuentra instalada en el primer puerto PCIe.

En nuestro caso concreto teníamos la siguiente configuración:

- Una **NVIDIA GeForce RTX 4060 Ti** instalada en el PCIe principal.
- Una **AMD Radeon RX 460/560** instalada en el PCIe secundario.
- La NVIDIA reservada completamente para **PCI Passthrough mediante vfio-pci**.
- La AMD destinada al escritorio, terminal, LightDM y servidor X del propio Proxmox.
- Monitores conectados físicamente a ambas tarjetas gráficas.

Al encender el equipo con los dos monitores conectados, la UEFI seleccionaba la NVIDIA como adaptador gráfico inicial. GRUB también aparecía a través de ella y el framebuffer creado por el firmware quedaba asociado inicialmente a esta tarjeta.

Impedir que Linux utilice el framebuffer creado por la UEFI

El primer paso consiste en **impedir que Linux registre y continúe utilizando el framebuffer que la UEFI creó sobre la NVIDIA**. Esto no cambia qué tarjeta utiliza la propia UEFI ni qué tarjeta utiliza GRUB. El cambio se produce cuando empieza a ejecutarse el kernel Linux.

Podemos comprobar primero nuestra línea actual de arranque:

```
cat /proc/cmdline
```

En nuestro caso obteníamos:

```
BOOT_IMAGE=/boot/vmlinuz-7.0.14-17-pve root=UUID=a34ac596-9c29-4c7b-98c4-5602f516cb3c ro net.ifnames=0
biosdevnames=0 amd_iommu=on iommu=pt pcie_acs_override=downstream,multifunction
```

Para evitar que el kernel inicialice el framebuffer del firmware añadimos el siguiente parámetro:

```
initcall_blacklist=sysfb_init
```

El parámetro **initcall_blacklist** permite indicarle al kernel que no ejecute determinadas funciones de inicialización. En este caso bloqueamos **sysfb_init**, evitando que el framebuffer proporcionado por el firmware sea registrado como dispositivo gráfico del sistema.

Si utilizamos GRUB, editamos:

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

Y añadimos el parámetro a nuestra línea existente. En nuestro caso quedó de la siguiente manera:

```
GRUB_CMDLINE_LINUX_DEFAULT="net.ifnames=0 biosdevnames=0 amd_iommu=on iommu=pt
pcie_acs_override=downstream,multifunction initcall_blacklist=sysfb_init"
```

Actualizamos GRUB:

```
update-grub
```

Después de reiniciar, observaremos un comportamiento interesante. **La UEFI y GRUB seguirán apareciendo en el monitor conectado a la NVIDIA**, ya que en esas etapas Linux todavía no tiene control sobre el hardware. Sin embargo, en cuanto el kernel comienza a cargarse, el framebuffer de la UEFI deja de utilizarse y la consola puede pasar automáticamente a la tarjeta AMD controlada por amdgpu.

En nuestro caso obtuvimos exactamente este comportamiento:

```
UEFI / BIOS  → NVIDIA
GRUB         → NVIDIA
Kernel Linux → AMD
Consola      → AMD
NVIDIA       → vfio-pci
```

Solucionar el fallo de LightDM y Xorg

Después del cambio anterior conseguimos que la terminal apareciese correctamente a través de la AMD, pero nos encontramos con un segundo problema: **LightDM no conseguía iniciar el servidor gráfico**.

Podemos revisar el estado de LightDM mediante:

```
journalctl -b -u lightdm --no-pager -n 200
```

Y posteriormente revisar sus registros:

```
cat /var/log/lightdm/lightdm.log
```

En nuestro caso LightDM intentaba iniciar Xorg, pero Xorg terminaba inmediatamente con código de error 1. El error real apareció al consultar:

```
cat /var/log/lightdm/x-0.log
```

El mensaje relevante era:

```
vesa: Refusing to run, Framebuffer or dri device present

Fatal server error:
Cannot run in framebuffer mode. Please specify busIDs for all framebuffer devices
```

Aunque el kernel ya estaba utilizando correctamente amdgpu, Xorg seguía encontrándose ante un sistema con varias tarjetas gráficas y no sabía inequívocamente cuál debía utilizar.

La solución consiste en **indicar explícitamente a Xorg el dispositivo PCI correspondiente a la tarjeta AMD**.

Obtener el BusID de la tarjeta gráfica AMD

Podemos identificar nuestras tarjetas gráficas mediante:

```
lspci -Dnnk | grep -A3 -E 'VGA compatible controller|3D controller|Display controller'
```

En nuestro sistema obteníamos:

```
0000:14:00.0 VGA compatible controller [0300]: NVIDIA Corporation AD106 [GeForce RTX 4060 Ti] [10de:2803] (rev a1)
        Subsystem: Micro-Star International Co., Ltd. [MSI] Device [1462:5174]
        Kernel driver in use: vfio-pci
        Kernel modules: nvidiafb, nouveau

0000:15:00.0 VGA compatible controller [0300]: Advanced Micro Devices, Inc. [AMD/ATI] Baffin [Radeon RX 460/560D /
Pro 450/455/460/555/555X/560/560X] [1002:67ef] (rev e5)
        Subsystem: Advanced Micro Devices, Inc. [AMD/ATI] Device [1002:0b04]
        Kernel driver in use: amdgpu
        Kernel modules: amdgpu
```

Esto confirma además que la configuración de passthrough está funcionando correctamente: **la NVIDIA está controlada por vfio-pci y la AMD por amdgpu**.

La AMD se encuentra en:

```
0000:15:00.0
```

Hay que tener en cuenta que lspci representa el número del bus PCI en **hexadecimal**. Xorg utiliza tradicionalmente su representación decimal dentro de BusID. En nuestro caso:

```
0x15 = 21
```

Por tanto, el identificador que utilizaremos para Xorg será:

```
PCI:21:0:0
```

Naturalmente, este valor será diferente en cada equipo, por lo que **no debemos copiar PCI:21:0:0 directamente sin comprobar previamente la dirección PCI de nuestra propia tarjeta gráfica**.

Forzar la tarjeta gráfica que utilizará Xorg

Antes de continuar podemos comprobar que tenemos instalado el controlador Xorg para amdgpu:

```
dpkg -l xserver-xorg-video-amdgpu
```

Creamos entonces el directorio de configuración si todavía no existe:

```
mkdir -p /etc/X11/xorg.conf.d
```

Y creamos el archivo:

```
nano /etc/X11/xorg.conf.d/10-amdgpu.conf
```

En nuestro caso introdujimos:

```
Section "Device"
  Identifier "AMD"
  Driver "amdgpu"
  BusID "PCI:21:0:0"
EndSection

Section "Screen"
  Identifier "AMD-Screen"
  Device "AMD"
EndSection
```

La sección Device identifica inequívocamente nuestra AMD y especifica que debe utilizarse el controlador amdgpu. La directiva **BusID** evita que Xorg tenga que decidir automáticamente entre las diferentes tarjetas PCI disponibles.

La sección Screen referencia posteriormente este dispositivo, haciendo que la sección Device quede activa cuando Xorg inicia el servidor gráfico.

Podemos probar la configuración sin reiniciar todo el servidor:

```
systemctl restart lightdm
```

Si la configuración es correcta, **LightDM debería aparecer directamente en el monitor conectado a la AMD**.

Finalmente reiniciamos el host manteniendo desde el principio los monitores conectados a las dos tarjetas gráficas:

```
reboot
```

Después de aplicar ambos cambios, el proceso completo de arranque queda de la siguiente manera:

```
Encendido
   │
   ▼
UEFI
   │
   └── NVIDIA
        │
        ▼
GRUB
   │
   └── NVIDIA
        │
        ▼
Kernel Linux
   │
   ├── sysfb_init bloqueado
   │
   ├── NVIDIA → vfio-pci
   │
   └── AMD    → amdgpu
                  │
                  ▼
               Consola
                  │
                  ▼
               LightDM
                  │
                  ▼
                Xorg
                  │
```

```
└── AMD
```

La NVIDIA queda así **sin ser utilizada por el sistema gráfico del host** y continúa disponible para PCI Passthrough. Podemos arrancar posteriormente nuestra máquina virtual y asignarle la tarjeta mediante vfio-pci sin que el framebuffer, LightDM o Xorg del host estén utilizando sus recursos.

En nuestro caso comprobamos finalmente que la máquina virtual seguía arrancando correctamente con la NVIDIA pasada directamente mediante PCI Passthrough, mientras que **la consola y el entorno gráfico de Proxmox permanecían completamente sobre la AMD**.

Debemos tener presente que este procedimiento **no cambia la tarjeta que utiliza la UEFI ni la que utiliza GRUB**. Si el firmware de nuestra placa base siempre inicializa la GPU situada en el primer PCIe, seguiremos viendo esas primeras etapas del arranque en el monitor conectado a dicha tarjeta. El cambio se produce en cuanto Linux comienza a ejecutarse y toma el control del hardware.