PCI Passthrough

Contents

Introduction

Note: This is a collection of examples, workarounds, hacks, and specific issues for PCI(e) passthrough. For a step-by-step guide on how and what to do to pass through PCI(e) devices, see the docs (https://pve.proxmox.com/pve-docs/pve-admin-guide.html#qm_pci_passthrough) or the wiki page generated from the docs

PCI passthrough allows you to use a physical PCI device (graphics card, network card) inside a VM (KVM virtualization only).

If you "PCI passthrough" a device, the device is not available to the host anymore. Note that VMs with passed-through devices cannot be migrated.

Requirements

This is a list of basic requirements adapted from the Arch wiki (https://wiki.archlinux.org/title/PCI_passthrough_via_OVMF#Prerequisites)

CPU requirements
Your CPU has to support hardware virtualization and IOMMU. Most new CPUs support this.

- AMD: CPUs from the Bulldozer generation and newer, CPUs from the K10 generation need a 890FX or 990FX motherboard.
- Intel: list of VT-d capable Intel CPUs (https://ark.intel.com/content/www/us/en/ark/search/featurefilter.html?productType=873&0_VTD=True)

Motherboard requirements
Your motherboard needs to support IOMMU. Lists can be found on the Xen wiki (https://wiki.xenproject.org/wiki/VTd_HowTo) and Wikipedia (https://en.wikipedia.org/wiki/List_of_IOMMU-supporting_hardware). Note that, as of writing, both these lists are incomplete and very out-of-date and most newer motherboards support IOMMU.

GPU requirements
The ROM of your GPU does not necessarily need to support UEFI, however, most modern GPUs do. If you GPU ROM supports UEFI, it is recommended to use OVMF (UEFI) instead of SeaBIOS. For a list of GPU ROMs, see Techpowerup's collection of GPU ROMs (https://www.techpowerup.com/vgabios/?architecture=&manufacturer=&model=&version=&interface=&memType=&memSize=&since=)

Verifying IOMMU parameters

Verify IOMMU is enabled

Reboot, then run:

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

There should be a line that looks like "DMAR: IOMMU enabled". If there is no output, something is wrong.

Verify IOMMU interrupt remapping is enabled

It is not possible to use PCI passthrough without interrupt remapping. Device assignment will fail with 'Failed to assign device "[device name]": Operation not permitted' or 'Interrupt Remapping hardware not found, passing devices to unprivileged domains is insecure.'.

All systems using an Intel processor and chipset that have support for Intel Virtualization Technology for Directed I/O (VT-d), but do not have support for interrupt remapping will see such an error. Interrupt remapping support is provided in newer processors and chipsets (both AMD and Intel).

To identify if your system has support for interrupt remapping:

```
dmesg | grep 'remapping'
```

If you see one of the following lines:

- `AMD-Vi: Interrupt remapping enabled`
- `DMAR-IR: Enabled IRQ remapping in x2apic mode` ('x2apic' can be different on old CPUs, but should still work)

then remapping is supported.

If your system doesn't support interrupt remapping, you can allow unsafe interrupts with:

```
echo "options vfio_iommu_type1 allow_unsafe_interrupts=1" > /etc/modprobe.d/iommu_unsafe_interrupts.conf
```

Verify IOMMU isolation

For working PCI passthrough, you need a dedicated IOMMU group for all PCI devices you want to assign to a VM.

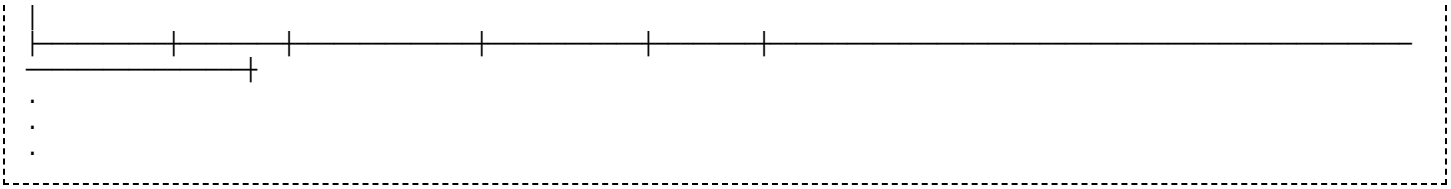
When executing

```
# pvesh get /nodes/{nodename}/hardware/pci --pci-class-blacklist ""
```

replacing {nodename} with the name of your node.

You should get a list similar to:

```
┌──────────┬────────┬──────────────┬─────────────┬────────┬──────────────────────────────────────────────────────────────┬...
│ class    │ device │ id           │ iommugroup  │ vendor │ device_name                                                  │
╞══════════╪════════╪══════════════╪═════════════╪════════╪══════════════════════════════════════════════════════════════╪═══
│ 0x010601 │ 0xa282 │ 0000:00:17.0 │           5 │ 0x8086 │ 200 Series PCH SATA controller [AHCI mode]                   │
├──────────┼────────┼──────────────┼─────────────┼────────┼──────────────────────────────────────────────────────────────┼───
│ 0x010802 │ 0xa808 │ 0000:02:00.0 │          12 │ 0x144d │ NVMe SSD Controller SM981/PM981/PM983                        │
├──────────┼────────┼──────────────┼─────────────┼────────┼──────────────────────────────────────────────────────────────┼───
│ 0x020000 │ 0x15b8 │ 0000:00:1f.6 │          11 │ 0x8086 │ Ethernet Connection (2) I219-V                               │
├──────────┼────────┼──────────────┼─────────────┼────────┼──────────────────────────────────────────────────────────────┼───
│ 0x030000 │ 0x5912 │ 0000:00:02.0 │           2 │ 0x8086 │ HD Graphics 630                                              │
├──────────┼────────┼──────────────┼─────────────┼────────┼──────────────────────────────────────────────────────────────┼───
│ 0x030000 │ 0x1d01 │ 0000:01:00.0 │           1 │ 0x10de │ GP108 [GeForce GT 1030]                                      │
```



To have separate IOMMU groups, your processor needs to have support for a feature called ACS (Access Control Services). Make sure you enable the corresponding setting in your BIOS for this.

If you don't have dedicated IOMMU groups, you can try moving the card to another PCI slot.

Should that not work, you can try using Alex Williamson's ACS override patch (https://lkml.org/lkml/2013/5/30/513). However, this should be seen as a last option and is not without risks (http://vfio.blogspot.be/2014/08/iommu-groups-inside-and-out.html).

As of writing, the ACS patch is part of the Proxmox VE kernel and can be invoked via Editing the kernel command line (https://pve.proxmox.com/pve-docs/chapter-sysadmin.html#sysboot_edit_kernel_cmdline). Add

```
pcie_acs_override=downstream
```

to the kernel boot command line (grub or systemd-boot) options.

More information can be found at Alex Williamson's blog (http://vfio.blogspot.com/).

GPU passthrough

Note: See http://blog.quindorian.org/2018/03/building-a-2u-amd-ryzen-server-proxmox-gpu-passthrough.html/ if you like an article with a How-To approach. (NOTE: you usually do not need the ROM-file dumping mentioned at the end!)

- AMD RADEON 5xxx, 6xxx, 7xxx, NVIDIA GeForce 7, 8, GTX 4xx, 5xx, 6xx, 7xx, 9xx, 10xx, 15xx, 16xx, and RTX 20xx have been reported working. Anything newer should work as well.
- AMD Navi (5xxx(XT)/6xxx(XT)) suffer from the reset bug (see https://github.com/gnif/vendor-reset), and while dedicated users have managed to get them to run, they require a lot more effort and will probably not work entirely stable (see the AMD specific issues for workarounds).
- You might need to load some specific options in grub.cfg or other tuning values to get your configuration specifically working/stable
- Here's a good forum thread of Arch Linux: https://bbs.archlinux.org/viewtopic.php?id=162768

For starters, it's often helpful if the host doesn't try to use the GPU, which avoids issues with the host driver unbinding and re-binding to the device. Sometimes making sure the host BIOS POST messages are displayed on a different GPU is helpful too. This can sometimes be acomplished via BIOS settings, moving the card to a different slot or enabling/disabling legacy boot support.

Blacklisting drivers

The following is a list of common drivers and how to blacklist them:

- AMD GPUs

```
echo "blacklist amdgpu" >> /etc/modprobe.d/blacklist.conf
echo "blacklist radeon" >> /etc/modprobe.d/blacklist.conf
```

- NVIDIA GPUs

```
echo "blacklist nouveau" >> /etc/modprobe.d/blacklist.conf
echo "blacklist nvidia*" >> /etc/modprobe.d/blacklist.conf
```

- Intel GPUs

```
echo "blacklist i915" >> /etc/modprobe.d/blacklist.conf
```

> **Note**: If you are using an Intel iGPU and an Intel discrete GPU, blacklisting the Intel 'i915' drivers that the discrete GPU uses means the iGPU won't be able to use those drivers either.

After blacklisting, you will need to reboot.

How to know if a graphics card is UEFI (OVMF) compatible

Have a look at the requirements section. Chances are you are using the BIOS listed for your device on the Techpowerup GPU ROM list, which will say if it is UEFI compatible or not.

Alternatively, you can dump your ROM and use Alex Williams rom-parser tool:

> **Note**: You will want to run the following commands logged in as root user (by running `su -`) or by wrapping them with `sudo sh -c "<code-snippet>"`, otherwise the bash-redirects in the code-snippets below won't work

Get and compile the software "rom-parser":

```
git clone https://github.com/awilliam/rom-parser
cd rom-parser
make
```

Then dump the rom of you vga card:

```
cd /sys/bus/pci/devices/0000:01:00.0/
echo 1 > rom
cat rom > /tmp/image.rom
echo 0 > rom
```

and test it with:

```
./rom-parser /tmp/image.rom
```

The output should look like this:

```
Valid ROM signature found @0h, PCIR offset 190h
 PCIR: type 0, vendor: 10de, device: 1280, class: 030000
 PCIR: revision 0, vendor revision: 1
Valid ROM signature found @f400h, PCIR offset 1ch
 PCIR: type 3, vendor: 10de, device: 1280, class: 030000
 PCIR: revision 3, vendor revision: 0
  EFI: Signature Valid
 Last image
```

To be UEFI compatible, you need a "type 3" in the result.

The 'romfile' option

Some motherboards can't pass through GPUs on the first PCI(e) slot by default, because its vBIOS is shadowed during boot up. You need to capture its vBIOS when it is working "normally" (i.e. installed in a different slot), then you can move the card to slot 1 and start the vm using the dumped vBIOS.

To dump the bios:

```
cd /sys/bus/pci/devices/0000:01:00.0/
echo 1 > rom
cat rom > /usr/share/kvm/vbios.bin
echo 0 > rom
```

Then you can pass the vbios file (must be located in /usr/share/kvm/) with:

```
hostpci0: 01:00,x-vga=on,romfile=vbios.bin
```

Tips

Some Windows applications like GeForce Experience, Passmark Performance Test and SiSoftware Sandra can crash the VM. You need to add:

```
echo "options kvm ignore_msrs=1" > /etc/modprobe.d/kvm.conf
```

If you see a lot of warning messages in your 'dmesg' system log, add the following instead:

```
echo "options kvm ignore_msrs=1 report_ignored_msrs=0" > /etc/modprobe.d/kvm.conf
```

Nvidia Tips

User have reported that NVIDIA Kepler K80 GPUs need this in vmid.conf:

```
args: -machine pc,max-ram-below-4g=1G
```

Troubleshooting

"BAR 3: can't reserve [mem]" error

If you have this error when you try to use the card for a VM:

```
vfio-pci 0000:04:00.0: BAR 3: can't reserve [mem 0xca000000-0xcbffffff 64bit]
```

you can try to add the following kernel command line option:

```
video=efifb:off
```

Check out the documentation about editing the kernel command line (https://pve.proxmox.com/pve-docs/chapter-sysadmin.html#sysboot_edit_kernel_cmdline).

WSLg (Windows Subsystem for Linux GUI)

If GUI apps don't open in WSLg, see Windows 2022 guest best practices (https://pve.proxmox.com/wiki/Windows_2022_guest_best_practices#Installing_WSL.28g.29).

Black display in NoVNC/Spice

If you are passing through a GPU and are getting a black screen, you might need to change your display settings in the Guest OS. On Windows, this can be done by pressing the "Super/Windows" and "P" key. Alternatively, if you are using the GPU for hardware accelerated computing and need no graphical output from it, you can deselect the "primary GPU" option and physically disconnect your GPU.

Spice

Spice may give trouble when passing through a GPU as it presents a "virtual" PCI graphic card to the guest and some drivers have problems with that, even when both cards show up. It's always worth a try to disable SPICE and check again if something fails.

HDMI audio crackling/broken

Some digital audio devices (usually added via GPU functions) may require MSI (Message Signaled Interrupts) to be enabled to function correctly. If you experience any issues, try changing MSI settings in the guest and rebooting the guest.

Linux guests usually enable MSI by themselves. To force use of MSI for GPU audio devices, use the following command and reboot:

```
echo "options snd-hda-intel enable_msi=1" >> /etc/modprobe.d/snd-hda-intel.conf
```

Use 'lspci -vv' and check for the following line on your device to see if MSI is enabled:

```
Capabilities: [a0] MSI: Enable+ Count=1/1 Maskable- 64bit+
```

If it says 'Enable+', MSI is working, 'Enable-' means it is supported but disabled, and if the line is missing, MSI is not supported by the PCIe hardware.

This can potentially also improve performance for other passthrough devices, including GPUs, but that depends on the hardware being used.

BIOS options

Make sure you are using the most recent BIOS version for you motherboard. Often IOMMU groupings or passthrough support in general is improved in later versions.

Some general BIOS options that might need changing to allow passthrough to work:

- IOMMU or VT-d: Set to 'Enabled' or equivalent, often 'Auto' is not the same
- 'Legacy boot' or CSM: For GPU passthrough it can help to disable this, but keep in mind that PVE has to be installed in UEFI mode, as it will not boot in BIOS mode without this enabled. The reason for disabling this is that it avoids legacy VGA initialization of installed GPUs, making them able to be re-initialized later, as required for passthrough. Most useful when trying to use passthrough in single GPU systems.
- 'Resizable BAR'/'Smart Access Memory': Some AMD GPUs (Vega and up) experience 'Code 43' in Windows guests if this is enabled on the host. It's not supported in VMs either way (yet), so the recommended setting is 'off'.

Error 43

Error code 43 (https://support.microsoft.com/en-us/windows/fix-graphics-device-problems-with-error-code-43-6f6ae1ec-0bbe-a848-142e-0c6190502842) is a generic Windows driver error and can occur for a wide number of reasons. Things you can try troubleshooting include:

Finding out if the PCI device has a hardware fault

- Try passing the PCI device to a Linux VM
- Try plugging the PCI device into a different PCI slot or into a different machine

Finding software issues

- Check the security event logs of your Windows VM
- Check the dmesg logs of your host machine
- Dump your vBIOS and check if it is working correctly.
- Try a different vbios (see the GPU requirements section)

- If your GPU supports resizable BAR/SAM and you have this option set in your BIOS, you might need to deactivate it or manually tweak your BAR using an udev rule (see Code 43 while Resizable Bar is turned on in the bios (https://wiki.archlinux.org/title/PCI_passthrough_via_OVMF#Code_43_while_Resizable_Bar_is_turned_on_in_the_bios) in the Arch wiki)
- Sometimes the issue is very hardware-dependent. You might find someone else who found a solution who has the same hardware. Try searching the internet with keywords containing your hardware, together with keywords like "Proxmox", "KVM", or "Qemu".

Nvidia specific issues

When passing through mobile- or vGPUs, it might be necessary to spoof the Vendor ID and Hardware ID as if the passed-through GPU were the desktop variant. Changing the IDs might also be needed to remove manufacturer-specific vendor ID variants that are not recognized otherwise.

The Vendor and Device ID can be added in the web interface under "Hardware" -> "PCI Device (hostpciX)" and then clicking on the "Advanced" checkbox.

Some software will also refuse to run when it detects that it is running in a VM. This should no longer be an issue with Nvidia drivers 465 and newer.

To find the Vendor ID and Device ID of the card installed on your host, run:

```
lspci -nn
```

which will give you something similar to

```
01:00.0 VGA compatible controller [0300]: NVIDIA Corporation GP108 [GeForce GT 1030] [10de:1d01] (rev a1)
```

Here, `0x10de` is the Vendor ID and `0x1d01` the Device ID.

AMD specific issues

Some AMD cards suffer from the "AMD reset bug" where the GPU does not correctly reset after power cycling. This can be remedied with the vendor-reset patch (https://github.com/gnif/vendor-reset/). See also Nick Sherlock's writeup (https://www.nicksherlock.com/2020/11/working-around-the-amd-gpu-reset-bug-on-proxmox/) on the issue.

USB passthrough

If you need to pass through USB devices (keyboard, mouse), please follow the USB Physical Port Mapping wiki article.

vGPU

If you want to split up one GPU into multiple vGPUs, see:

- MxGPU with AMD S7150 (https://pve.proxmox.com/wiki/MxGPU_with_AMD_S7150_under_Proxmox_VE_5.x)

- NVIDIA vGPU (https://pve.proxmox.com/wiki/NVIDIA_vGPU_on_Proxmox_VE_7.x)
