qemu-web-desktop: GPU Installation

You should have already installed the qemu-web-desktop/DARTS from INSTALL.md. This documentation assumes you have some GPU to pass to the service.

:warning: The specified GPU will be "extracted" from the host system, and only usable via virtualization. It is then highly recommended to first make sure your hardware provides more than one display driver. :warning: This feature is NOT supported on Apple/MacOS (Metal).

Table of contents:

1. Installation: GPU pass-through
2. Manual GPU configuration
3. Common issues with GPU pass-through
    - VFIO not attached
    - Memory error (vfio_dma)
    - IOMMU groups (group not viable)

Installation: GPU pass-through

It is possible to use a physical GPU inside virtual machine sessions.

:warning: This GPU is exclusively attached to the virtual machine, and can not anymore be used on the server for display. This implies that you should have at least two distinct GPU's (of different model).

In the following, we assume we have a server with an AMD CPU, and NVIDIA GPU's, all running on a Debian system. The first step is to ensure that your server can detach a GPU from the host system. The feature which is used is called IOMMU/VFIO.

```
sudo dmesg | grep "AMD-Vi\|Intel VT-d"
[    1.059323] AMD-Vi: IOMMU performance counters supported
lscpu | grep -i "Virtualisation"
Virtualisation :                        AMD-V
egrep -q '^flags.*(svm|vmx)' /proc/cpuinfo && echo virtualization extensions available
virtualization extensions available
lspci -nnv | grep -i "VGA\|Audio\|3d controller\|Kernel driver in use: snd_hda_intel\|Kernel
4c:00.0 VGA compatible controller [0300]: NVIDIA Corporation GP108 [10de:1d01] (rev a1) (pro
    Kernel driver in use: nvidia
4c:00.1 Audio device [0403]: NVIDIA Corporation GP108 High Definition Audio Controller [10de
    Subsystem: ASUSTeK Computer Inc. GP108 High Definition Audio Controller [1043:8746]
    Kernel driver in use: snd_hda_intel
4d:00.0 VGA compatible controller [0300]: NVIDIA Corporation GP108 [10de:1d01] (rev a1) (pro
    Kernel driver in use: nvidia
4d:00.1 Audio device [0403]: NVIDIA Corporation GP108 High Definition Audio Controller [10de
    Subsystem: ASUSTeK Computer Inc. GP108 High Definition Audio Controller [1043:8621]
    Kernel driver in use: snd_hda_intel
```

which results in a list of available GPU. In the following, we assume we have two low-cost/power NVIDIA GT 1030 (384 cores, 2 GB memory) cards, on PCI addresses `4c:00` and `4d:00`. It is important to also take note of the hardware vendor:model code for the GPU, here `10de:1d01` and `10de:0fb8` for the video **and** audio parts.

In short An automatic configuration is achieved e.g. via the command:

```
sudo qwdctl gpu 10de:1d01
```

The `sudo qwdctl gpu` command without the GPU IDs will show the available GPUs and request IDs. The command requests for confirmation, but you may add the `--yes` for a fully automatic configuration.

The GPU-pass-through section is uncommented in the index.html file in the `html/desktop` directory.

The configuration option `service_max_gpu_nb_per_user` (default 1) allows to limit the number of GPU assigned to one user.

To unlock/re-attach all virtualized GPU's to the server (uninstall pass-through), enter command:

```
sudo qwdctl gpu_unlock
```

which also supports the `--yes` option.

Manual GPU configuration

In case you prefer to configure the GPU manually, step by step, read further.

The files that need customization are the following:

```
sudo mkdir -p /etc/systemd/system/apache2.service.d
sudo gedit /etc/default/grub /etc/modprobe.d/vfio.conf /etc/initramfs-tools/modules /etc/ude
```

In the following step, we detach these GT 1030 cards at boot. In the `/etc/default/grub` file activate IOMMU, and flag the vendor:model codes (here with video and sound parts - multiple cards are possible separated with commas):

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet amd_iommu=on iommu=pt vfio-pci.ids=10de:1d01,10de:0fb8"
```

For Intel CPU's, you would use option `intel_iommu=on`. This GPU information should also be added as a `modprobe` option. Create for instance the file `/etc/modprobe.d/vfio.conf` with content:

```
# /etc/modprobe.d/vfio.conf
options vfio-pci ids=10de:1d01,10de:0fb8 disable_vga=1
```

and push necessary modules into the kernel by adding:

```
# /etc/initramfs-tools/modules
vfio
vfio_iommu_type1
vfio_pci
vfio_virqfd
vhost-netdev
```

into file `/etc/initramfs-tools/modules`.

Finally reconfigure the boot and linux kernel, and restart the server:

```
sudo update-initramfs -u
sudo update-grub
sudo reboot
```

After reboot, the command `lspci -nnk` will show the detached cards as used by the `vfio-pci` kernel driver.

```
4d:00.0 VGA compatible controller [0300]: NVIDIA Corporation GP108 [GeForce GT 1030] [10de:1
    Subsystem: ASUSTeK Computer Inc. GP108 [GeForce GT 1030] [1043:8621]
    Flags: fast devsel, IRQ 4, IOMMU group 61
    Memory at ad000000 (32-bit, non-prefetchable) [disabled] [size=16M]
    Memory at 70000000 (64-bit, prefetchable) [disabled] [size=256M]
    Memory at 80000000 (64-bit, prefetchable) [disabled] [size=32M]
    I/O ports at a000 [disabled] [size=128]
    Expansion ROM at ae000000 [disabled] [size=512K]
    Capabilities: <access denied>
    Kernel driver in use: vfio-pci
    Kernel modules: nvidia
```

:warning: all identical GPU of that model (`10de:1d01`) are detached. It is not possible to keep one on the server, and send the other same model to the VM. This is why at least two different GPU models are physically needed in the computer.

It is now necessary to configure the system so that the Apache user can launch qemu with IOMMU/VFIO pass-through. Else you get errors such as:

```
VFIO: ... permission denied
```

Change VFIO access rules so that group `kvm` can use it. Add in file `/etc/udev/rules.d/10-qemu-hw-users.rules`:

```
# /etc/udev/rules.d/10-qemu-hw-users.rules
SUBSYSTEM=="vfio", OWNER="root", GROUP="kvm"
```

then restart `udev`

```
sudo udevadm control --reload-rules
sudo udevadm trigger
```

Edit the `/etc/security/limits.conf` file and add at the end:

```
# /etc/security/limits.conf
*    soft memlock 20000000
*    hard memlock 20000000
@kvm soft memlock unlimited
@kvm hard memlock unlimited
```

Customized the Apache start-up with:

```
sudo mkdir -p /etc/systemd/system/apache2.service.d/
sudo nano /etc/systemd/system/apache2.service.d/override.conf
```

and enter:

```
# /etc/systemd/system/apache2.service.d/override.conf
[Service]
LimitMEMLOCK=infinity
```

Then update the boot process:

```
sudo systemctl daemon-reload
sudo systemctl restart apache2
```

Make sure the GPU-pass-through section is uncommented in the index.html file in the `html/desktop` directory.

The configuration option `service_max_gpu_nb_per_user` (default 1) allows to limit the number of GPU assigned to one user.

Then, for testing purposes, you may launch a `qemu` command, such as:

```
qemu-system-x86_64  -m 4096 -smp 4 -hda debian10.qcow2 -name Debian -device ich9-ahci,id=ahc
```

You may specify a list of black-listed GPU in the `$config{gpu_blacklist}` item, to e.g. reserve a GPU for other purposes, or isolate a defective device.

Common issues with GPU pass-through

VFIO not attached In some cases, the GPU is still not handled by the VFIO driver, as show with `lspci -nnvk`:

```
Kernel driver in use: nouveau
Kernel modules: nouveau
```

If the GPU is not sent to the `vfio-pci` driver, look into the `dmesg` output for the GPU ID/PCI, to see if some error occurs.

You may first manually attach the GPU to the VFIO by issuing the following commands (adapt the `4d:00.0` PCI address to yours):

```
echo vfio-pci > /sys/bus/pci/devices/0000:4d:00.0/driver_override
echo 0000:4d:00.0 > /sys/bus/pci/drivers_probe
```

Then check again the `lspci -nnvk` for your device.

You may also try to install the vendor drivers (e.g. NVIDIA or AMD) for the GPU on the server, to see if this solves the issue.

Memory error (vfio_dma) When running the web service, you may experience in the Apache `/var/log/apache2/error.log` messages like:

```
qemu-system-x86_64: -device vfio-pci,host=0000:4c:00.0,multifunction=on: VFIO_MAP_DMA: -12
qemu-system-x86_64: -device vfio-pci,host=0000:4c:00.0,multifunction=on: vfio_dma_map(0x5596
```

as well as:

```
vfio_pin_pages_remote: RLIMIT_MEMLOCK (65536) exceeded
```

in `dmesg` which is triggered by a low memory allocation threshold `ulimit`.

Adapt the memory pre-allocation for the GPU. This is done in `/etc/security/limits.conf` by adding lines at the end:

```
# /etc/security/limits.conf
*    soft memlock 20000000
*    hard memlock 20000000
@kvm soft memlock unlimited
@kvm hard memlock unlimited
```

The value is given in Kb, here 20 GB for all users, and unlimited for group `kvm`. Perhaps this 20 GB value should match the internal GPU memory.

Do something similar when Apache starts with SystemD e.g. in `/etc/systemd/system/multi-user.target.wan`

```
# /etc/systemd/system/multi-user.target.wants/apache2.service
[Service]
...
LimitMEMLOCK=infinity
```

It is also possible (and recommended) to configure the Apache service without modifying the whole systemd script. Just use:

```
sudo systemctl edit apache2.service
```

and enter the content of the 'override' file /etc/systemd/system/apache2.service.d/override.conf

```
# /etc/systemd/system/apache2.service.d/override.conf
[Service]
LimitMEMLOCK=infinity
```

IOMMU groups (group not viable) The GPU are attached to physical PCI connectors, which arrangement is handled by the system with a topology seen in the IOMMU groups. But, in order for QEMU/KVM to pass-through a device (GPU), it must be bound to a single IOMMU. In case the GPU is part of an IOMMU with other stuff in, *all* these must also be detached via the VFIO driver.

Then you will see error messages such as

```
group 60 is not viable
Please ensure all devices within the iommu_group are bound to their vfio bus driver.
```

First check that indeed your GPU are not alone in their IOMMU group. The following command displays the IOMMU groups and the attached devices.

```
for d in /sys/kernel/iommu_groups/*/devices/*; do n=${d#*/iommu_groups/*}; n=${n%%/*}; print
```

The first step is to make sure that you have included both the video *and* audio parts from the GPU. These usually go together in the same IOMMU group, and thus should be attached to `vfio-pci` in files `/etc/default/grub` and `/etc/modprobe.d/vfio.conf`. In case the faulty IOMMU group also contains other components, you may decide to add them into these two grub/modprobe files as well.

If the issue persists, you may physically move the GPU cards to other PCI-slots in order to find better arrangements. But this is not always effective, nor possible.

You can further allow your BIOS to shuffle a little the IOMMU groups with the settings (for AMD CPU's), e.g.:

- mode NUMA BIOS/AMD CBS/DF/Memory addressing/NBS4
- mode BIOS/AMD CBS/NBIO/PCIe ARI=Enabled
- mode BIOS/AMD CBS/NBIO/IOMMU=Enabled

Last, when all this fails, a definitive solution for kernels below 6.x is to use a special patch for the Linux kernel, known as `pcie_acs_override`. You will need to use a special kernel from e.g. https://liquorix.net/#install Make sure you get the same Linux kernel version as the one you currently run, so that GPU drivers (NVIDIA) are compatible. On a Debian system, you would for instance add `deb http://liquorix.net/debian bullseye main` to `/etc/apt/sources.list`, and issue:

```
sudo apt install linux-headers-5.10.0-17.1-liquorix-amd64 linux-image-5.10.0-17.1-liquorix-a
```

And finally add into the `/etc/default/grub`

```
GRUB_CMDLINE_LINUX_DEFAULT= ... pcie_acs_override=downstream,multifunction
```

and reboot. Now there should be one IOMMU group per device.