GPU Passthrough mit KVM

GPU-Passthrough mit KVM
Einleitung

Diese Anleitung zeigt, wie eine VM erstellt wird, der anstatt einer virtuellen eine durchgereichte Grafikkarte zur Verfügung steht. Als Virtualisierer wird KVM verwendet. Als Gastsystem wird Windows 7 verwendet.

Virtualisierung ist heutzutage nichts besonderes. Will man allerdings GPU-intensive Anwendungen in der Virtuellen Maschine ausführen, reicht die Leistung der virtuellen Grafikkarte nicht mehr aus. Hier kommt GPU-Passthrough ins Spiel.

Vorteile der Virtualisierung mit GPU-Passthrough:

- Starten, Runterfahren, Anhalten der VM
- Native GPU-Leistung in VM
- Benutzen der nativen Treiber in VM
- Kein Neustart für ein Windows-Spielchen zwischendurch

Voraussetzungen

Voraussetzung für das Durchreichen einer GPU ist das Vorhandensein des VT-d-fähigen Systems.

Das sind bei Intel die 7er-Chipsätze, das Z68-Chipsatz und einige andere mit 6er-Chipsatz.
Das Mainboard muss die VT-d-Funktion im BIOS richtig implementiert haben. VT-d muss im BIOS eingeschaltet sein.
Die Asrock-Boards sind ganz gut dabei.

http://www.asrock.com/support/note/vt-d.pdf
Die CPU muss ebenfalls VT-d unterstützen.

Das sind bei Intel die Sockel-2011 CPUs und die „nicht-K-CPUs“ bei Sockel-1155.
Siehe auch:
http://ark.intel.com/search/advanced...ssors&VTD=true

Hostsystem

Das Host-System hat folgende Komponenten:

Intel i7-3820

Asrock X79 Extreme6 mit VT-d Unterstützung
16 GB Ram
Xfx AMD HD6950 2GB
Msi AMD HD5450 1GB
Asrock hat momentan eine gute Auswahl an Platinen mit VT-d-Unterstutzung.

Als Host-Betriebssystem wird 64bit-Gentoo mit Kernel 3.6.0 verwendet.

Zusätzlich wurde qemu, libvirt und virt-manager installiert.

Kerneloptionen

Im Kernel werden, falls ein eigener Kernel kompiliert wird, folgende Optionen aktiviert:

KVM-Optionen (Bild kernelOptionen.png)

CONFIG_KVM_CLOCK=y
CONFIG_KVM_GUEST=y
CONFIG_HAVE_KVM=y
CONFIG_HAVE_KVM_IRQCHIP=y
CONFIG_HAVE_KVM_EVENTFD=y
CONFIG_KVM_APIC_ARCHITECTURE=y
CONFIG_KVM_MMIO=y

CONFIG_KVM_ASYNC_PF=y
CONFIG_HAVE_KVM_MSI=y
CONFIG_KVM=m
CONFIG_KVM_INTEL=m
CONFIG_KVM_MMU_AUDIT=y

Virtualisierungsoptionen

CONFIG_PARAVIRT_GUEST=y
CONFIG_PARAVIRT_TIME_ACCOUNTING=y
CONFIG_PARAVIRT=y
CONFIG_PARAVIRT_CLOCK=y
CONFIG_VIRT_TO_BUS=y
CONFIG_VIRTIO_BLK=m
CONFIG_SCSI_VIRTIO=m
CONFIG_VIRTIO_NET=m
CONFIG_VIRTIO_CONSOLE=m
CONFIG_HW_RANDOM_VIRTIO=m
CONFIG_FB_VIRTUAL=m
CONFIG_VIRTIO=m
CONFIG_VIRTIO_RING=m
CONFIG_VIRTIO_PCI=m
CONFIG_VIRTIO_BALLOON=m
CONFIG_VIRTIO_MMIO=m
CONFIG_VIRTIO_MMIO_CMDLINE_DEVICES=y
CONFIG_VIRT_DRIVERS=y
CONFIG_VIRTUALIZATION=y

Der „PCI Stub driver“ wird ebenfalls aktiviert.

Erzeugen der VM

Die Module kvm und kvm_intel müssen geladen sein.

Der Dienst libvirtd muss aktiv sein, ggf. starten mit:

Code:

```
/etc/init.d/libvirtd restart
```

Anschließend wird der „Virtual Machine Manager“ gestartet. Bild vmm.png
Nach Klick auf „Neue virtuelle Machine erstellen“ öffnet sich der Wizard. (Bild vmmWizard.png)

In wenigen Schritten wird nun eine virtuelle Maschine erstellt.

Schritt 1

Namen der VM und Installationsart eingeben.

Schritt 2

Pfad zum Installationsmedium eingeben.

Schritt 3

RAM und CPUs einstellen.

2 GB Ram und 2 CPUs sollten erstmal genügen. Das kann später geändert werden.
Schritt 4

Speicherplatz für VM auswählen.

Entweder Plattenabbild erzeugen oder, zwecks besserer Leistung, extra Partition zuweisen. LVM ist auch möglich.
Schritt 5 und 6

Einstellungen kontrollieren und „vmvga“ oder „qxl“ als Video einstellen.

Nun kann Windows 7 installiert werden.

Nach der Installation wird der SVGAII- oder der Qxl-Treiber installiert.
Gleich danach wird die virtuelle Grafikkarte im Gerätemanager deaktiviert, sonst gibt es einen Ressourcenkonflikt. Eventuell ist ein Neustart notwendig.
Nun in den Einstellungen der VM auf die „Hardware hinzufügen“, dann "PCI Host Device" klicken.

Grafikkarte auswählen und auf „Abschliessen“ klicken. (Bild addHW.png)

Die Grafikkarte taucht nun in Windows-VM auf und der AMD-Treiber kann jetzt installiert werden.

Anmerkungen

Sowohl die HD6950, als auch die HD5450 lassen sich durchreichen.

Natürlich darf die durchgereichte Grafikkarte nicht durch den X-Server benutzt werden.
Falls doch, muss in xorg.conf die andere Grafikkarte angegeben werden.
xorg.conf braucht nur aus wenigen Zeilen zu bestehen.

Code:

```
Section "Device"
       Identifier  "Card0"
       BusID       "PCI:1:0:0"
EndSection
```

Die HD6950 had PCI-ID 2:0:0, die HD5450 die 1:0:0. Ohne Änderung in xorg.conf wird die HD6950 vom X-Server benutzt.

Ich hatte noch das Problem, das die Grafikkarte bereits durch den Radeon-Treiber initialisiert wurde. Das gefällt dem Windows-Treiber nicht.

Meine Lösung:

Die Grafikkarte dem pci-stub-Treiber zuweisen:

Code:

```
echo "1002 6719" > /sys/bus/pci/drivers/pci-stub/new_id

echo "0000:02:00.0" > /sys/bus/pci/devices/0000\:02\:00.0/driver/unbind

echo "0000:02:00.0" > /sys/bus/pci/drivers/pci-stub/bind

echo "1002 aa80" > /sys/bus/pci/drivers/pci-stub/new_id

echo "0000:02:00.1" > /sys/bus/pci/devices/0000\:02\:00.1/driver/unbind

echo "0000:02:00.1" > /sys/bus/pci/drivers/pci-stub/bind
```

Dann den Rechner in den Standbymodus schalten und wieder einschalten.

Die Grafikkarte bleibt uninitialisiert.

Vielleicht gibt es eine elegantere Möglichkeit dies zu lösen, da ich aber immer den Standbymodus benutze, ist es für mich ein kleineres Problem.

Hier noch die Grub Zeile für Kernel:

linux /boot/kernel-3.6 root=/dev/sda1 raid=noautodetect intel_iommu=on fbcon=map:1 radeon.pcie_gen2=1

"fbcon=map:1" wird nur benötigt, falls die virtuellen consolen auf der 2. Grafikkarte erscheinen sollen.
Das Durchreichen funktioniert auch für Windows 8-, Windows XP- und Ubuntu 12-Gäste.

Links

http://www.linux-kvm.org/page/How_to...th_VT-d_in_KVM