Tutorial: Jessie: Virtuelles Windows mit 3D-Beschleunigung

Beitrag von NAB » 28.05.2015 17:39:07

1) Vorwort

Ab Abschnitt 4) höre ich mit dem Geschwafel auf und es geht ans Praktische!


2) Einführung

Hier geht es um das Virtualisieren von Windows. Das ist eigentlich nichts Neues, das kennt fast jeder von und mit Virtualbox. Fast jeder kennt auch gute Gründe dafür - sei es ein "original" Word, Photoshop, oder ein Spiel, das unter Wine nicht läuft. Bei Photoshop-Filtern und gerade bei Spielen stößt man aber auch schnell an die Grenzen - es fehlt die richtige 3D-Beschleunigung. Genau darum geht es hier - um ein virtuelles Windows, das eine echte Grafikkarte benutzt. So ein Setup kann das berüchtigte "Multiboot" fast vollständig ersetzen und man hat den Vorteil, dass beide Betriebssystem gleichzeitig laufen und z.B. über Samba Dateien austauschen können.

Zur Verwendung kommt hierbei Debian 8 - Jessie, 64 Bit, - möglichst ohne Fremdpakete.

Um den Text einigermaßen kurz zu halten, erwähne ich hier nicht jeden kleinen Schritt. Es existiert noch ein altes Tutorial von mir mit anfängertauglichen Beschreibungen - das ist zwar inzwischen veraltet, erklärt aber die kleinen Handgriffe besser. Das findet sich hier:

viewtopic.php?f=25&t=140895

a) Wissenswertes

Das Durchreichen von echter Hardware an eine VM (virtuelle Maschine) nennt man "passthrough". Genauer gesagt "pci-passthrough", wenn es um eine PCI- oder PCIe-Karte geht. Wenn es um eine Grafikkarte geht, nennt man es "vga-passthrough".
Die beste kostenfreie/quelloffene Lösung hierfür ist "qemu". Von qemu wurde vor einigen Jahren "kvm" abgespalten, um aus dem Emulator qemu einen Virtualisierer zu entwickeln. Inzwischen sind die Änderungen von kvm wieder zu qemu zurückgeflossen und das Gesamtpaket heißt nun wieder "qemu". Übrig geblieben ist nur das Kernelmodul "kvm", welches zur Virtualisierung benötigt wird.

"qemu" ist ein Komandozeilenprogramm, und die Definition einer VM kann schnell mal einige Zeilen lang und sehr unhandlich werden.

Für erste Experimente ist qemu trotzdem zu empfehlen, gerade wenn es Probleme gibt, um gezielt minimale Konfigurationen durchtesten zu können. Es finden sich per Google massenweise Beispiele für solche qemu-Zeilen, von denen man sich inspirieren lassen kann.

Für komplexere VMs empfiehlt sich das Framework "libvirt". Es verwaltet die Definitionen von VMs in XML-Dateien und lässt sich über "virsh", die "virtual shell", bedienen.

Auf libvirt wiederum setzt der "virt-manager" auf - eine graphische Oberfläche ähnlich Virtualbox.

Entwicklungsbedingt hinken die Fähigkeiten von libvirt immer den Konfigurationsmöglichkeiten von qemu hinterher. Ebenso hinkt die Entwicklung von virt-manager immer den Fähigkeiten von libvirt hinterher.

qemu kann zwei verschiedene Rechnerarchitekturen emulieren: i440FX und Q35. i440FX ist ein Intel-Chipsatz vom Ende der 90er und beherrscht nur PCI-Anschlüsse, während Q35 von ca. 2005 stammt und schon einen PCIe-Bus emulieren kann. In der Praxis funktioniert der i440FX aber auch unter neusten Windows-Versionen erstaunlich gut und wird von libvirt und virt-manager besser unterstützt. Vorteile von Q35 konnte ich bisher nicht entdecken, aber es soll einige wenige PCIe-Karten geben, die sich nur mit der Q35-Emulation erfolgreich durchreichen lassen.

"vga-passthrough" ist ein sich schnell entwickelndes experimentelles Feld. Die alte Methode hieß "pci-assign" und ist mittlerweile wieder eingestampft worden. Die aktuelle Methode heißt "vfio-pci". Genau so heißt auch das dafür verwendete Kernelmodul. Veraltete Tipps und Tutorials sind mit Vorsicht zu genießen.

b) Was geht - was geht nicht?

Ob es überhaupt geht, hängt von eurer Hardware ab - dazu mehr weiter unten. Wenn es geht, steht Windows eine echte Grafikkarte mit voller 3D-Beschleunigung zur Verfügung. Man spricht von Emulationsverlusten von ca. 10% ... das virtuelle Windows erreicht also ca. 90% der Leistung, die es direkt auf der Hardware hätte.
Was bisher nicht geht ist das Durchreichen der einzigen Grafikkarte. Man braucht also mindestens 2 Stück - üblicherweise eine Onboard-Grafik und eine separate Grafikkarte. Was ebenfalls nicht geht sind Hybridgrafiken wie "Optimus" - hier existieren zwar zwei Grafikkarten, die hängen aber beide am gleichen Framebuffer, und dürfen daher nur abwechselnd betrieben werden. Notebook-Besitzer dürften also größtenteils in die Röhre gucken. Was auch nicht geht ist das Durchreichen einer internen Intel-Grafik. An diesem Problem wird halbherzig gearbeitet - der Bedarf ist wohl gering.

Da hier zwei getrennte Grafikkarten werkeln, geht es natürlich nicht, beide über den gleichen Monitor-Anschluss zu betreiben. Ein

Umschalter hilft da, oder ein Monitor mit zwei umschaltbaren Eingängen, oder halt zwei Monitore.

Da Windows die Grafikkarte komplet unter seiner Fuchtel hat, kann Windows die Karte zum Beispiel übertakten oder ihre Lüfter steuern. HDMI mit Verschlüsselung und Ton laufen ebenso. Der gesamte Rest des PCs ist aber virtualisiert oder emuliert - da kommt Windows nicht dran. Ob "Blu Ray Wiedergabe" klappt, habe ich nicht getestet - ggf. muss da noch ein Sata-Controller mit eigenem Blu-Ray-Laufwerk an Windows durchgereicht werden.

Von Windows XP bis Windows 10 gibt es Erfolgsmeldungen, egal ob 32 oder 64 Bit. Unterhalb von Windows XP dürfte es schon mit geeigneten Treibern für die Grafikkarte schwer werden, außerdem benutzen Win98 & Co. noch kein HALT Kommando im Kernel, die CPU wird also immer bis zum Anschlag ausgelastet.

Technisch bedingt lassen sich nur PCIe-Steckplätze durchreichen. Technisch bedingt sind allerdings PCI-Steckplätze auf dem Mainboard an den PCIe-Bus angeschlossen und lassen sich daher auch durchreichen, allerdings nur alle auf einmal. Mit alten AGP-Karten dürfte man keine Chance haben.

c) PROBLEME

Probleme gibt es reichlich!

Das fängt bei "VGA" an. VGA ist eine Hardware-Erweiterung jeder Grafikkarte und war nie dafür vorgesehen, mehr als einmal innerhalb eines PCs aktiviert zu werden. Genau das passiert aber, wenn eine Grafikkarte an eine VM durchgereicht wird - die VM aktiviert beim Booten von der zweiten Grafikkarte deren VGA. Als Trick, um darum herum zu kommen, benutzen wir eine virtuelle Grafikkarte, die qemu bereit stellt, um von dieser zu booten. Die echte Grafikkarte wird als sekundäre Grafikkarte genutzt. Dabei wir ihr VGA nicht aktiviert. Das hat den Nebeneffekt, dass man noch ein Fenster mit Windows-Desktop auf seinem Debian-Desktop hat, was für den privaten Gebrauch kein Problem ist und durchaus praktisch sein kann.

Ebenso erwartet ein Betriebssystem beim Booten eine frische unbenutzte Grafikkarte. Das ist bei einer VM nur beim ersten Start der Fall. Beim zweiten Start findet sie die alte, benutzte Grafikkarte vor - denn die durchläuft ja keinen kompletten Neustart. Der neue vfio-Treiber sollte inzwischen ein Reset der durchgereichten Geräte ermöglichen. Das klappt aber nicht bei allen Grafikkarten gleich gut.

Weitere Probleme machen die Virtualisierungsfunktionen eures Mainboards und Prozessors. Es wimmelt vor fehlerhaften Implementierungen und unklaren Beschreibungen. Ein BIOS-Update kann da helfen, oder ein neuerer Kernel.

Bei den Grafikkarten ist auch nicht klar, welche sich problemlos durchreichen lassen. Hier hilft es nur, Google nach Erfolgsmeldungen zu befragen. Erfahrungsgemäß geht es mit AMD Grafikkarten besser. Nvidia arbeitet aktiv dagegen an, dass seine Karten durchgereicht

werden können, und die Entwickler von qemu müssen Wege drum herum finden. Das ist ein Katz und Maus Spiel.

Und dann ist da noch Windows. Windows ist leider nicht Windows, unterschiedliche Versionen verhalten sich unterschiedlich und benutzen unterschiedliche Treiber. Und diese Treiber haben unterschiedliche Fehler und Macken, je nach Version. Ich habe mir schon Konfigurationen gebastelt, die mit einem Windows frisch von der DVD problemlos liefen, und sich nach dem Installieren sämtlicher Windows-Updates als unbenutzbar erwiesen. Und ich habe natürlich keine Ahnung, welches der 137 Updates nun das störende war.

Bei so vielen Fehlerquellen ist es unmöglich, eine Anleitung zu schreiben, die bei allen funktioniert. Da hilft nur "selber herumprobieren".

3) Die Hardware

a) Ohne Hardware-Unterstützung geht nichts!

Das fängt bei der allgemeinen Virtualisierungs-Unterstützung an. Bei Intel heißt die "vt-x" und bei AMD "AMD V". Ohne läuft weder kvm noch Virtualbox. In /pcro/cpuinfo sollte "vmx" oder "svm" auftauchen.

Außerdem muss die Hardware "IOMMU" können. Eine MMU (Memory Management Unit) kann einen Speicherbereich in einen anderen Speicherbereich hineinblenden, so dass der Prozessor nicht dauernd Adressen umrechnen muss. Das nutzt jedes moderne Betriebssystem, um jedem Prozess seinen eigenen geschützten Speicherbereich zuzuweisen. Eine IOMMU macht genau das gleiche für "DMA", also für die Speicherbereiche, auf die PCI(e)-Geräte direkten I/O-Zugriff haben, um Daten besonders schnell und selbstständig kopieren zu können. Nur mit einer IOMMU ist es möglich, VMs direkten aber abgeschotteten Zugriff auf einzelne Geräte zu erlauben. Und ohne Hardware-Unterstützung ist das so übel langsam, dass es keiner benutzen wollen würde.

Die Technik gibt es erst seit 2009. Auf älteren Systemen hat mal also pauschal keine Chance.

Die IOMMU-Unterstützung heißt bei Intel "vt-d" und bei AMD "AMD-Vi". Bei beiden gilt, dass sowohl Prozessor als auch Chipsatz als auch BIOS die IOMMU unterstützen müssen.

Bei Intel-Prozessoren ist es noch relativ leicht - Intel unterhält eine vollständige Liste aller unterstützenden Prozessoren:

http://ark.intel.com/de/search/advanced?VTD=true Fü AMD finde ich sowas leider nicht, da muss man bei jedem Prozessor einzeln recherchieren. Pauschal gilt, dass IOMMU immer noch als "Premium-Feature" gesehen wird, und eher in den teureren Prozessoren zu finden ist. AMD ist da großzügiger als Intel.

Bei dem Mainboard kommt es auf den Chipsatz an.

Bei Intel können aktuelle Sockel-1150-Mainboards mit B85 oder Z87/Z97 Chipsatz auf jeden Fall IOMMU. Zuwiderlaufende Angaben beziehen sich nur auf die Intel-Grafik, die man aktuell eh nicht durchreichen kann. Für ältere Systeme schafft dieser Artikel etwas Durchblick (oder das Gegenteil davon):

http://www.heise.de/ct/hotline/System-m ... 55739.html Bei AMD ist es schon deswegen übersichtlicher, weil AMD in letzter Zeit weniger Chipsätze auf den Markt gebracht hat. Diese Liste gibt etwas Übersicht:
http://en.wikipedia.org/wiki/List_of_IO ... #AMD_based
Beim BIOS/UEFI wird es wiederum schwierig. Hier steht es jedem Hersteller frei, IOMMU freizuschalten oder auch nicht, und dabei beliebig viele Fehler zu machen, und die Funktion beim nächsten BIOS-Update wieder zu sperren, oder auch nicht. ASRock versucht, wo möglich, IOMMU pauschal freizuschalten:

http://wiki.xenproject.org/wiki/VTd_HowTo Bei Asus, MSI und Gigabyte muss man nachfragen ... oder per Google nach Erfolgsberichten suchen.

b) Zusatz-Hardware

Zusätzliche Hardware kann Probleme machen. Besonders unangenehm aufgefallen sind mir bisher Sata-Controller mit Marvell-Chipsatz - da ist praktisch jeder kaputt und muss erst per Kernel-Sonderbehandlung zum Laufen gebracht werden, was locker mal ein Jahr dauern kann. Besonders ärgerlich ist das, wenn der Chip fest auf dem Mainboard verlötet ist - da kann man nur hoffen, dass er sich im BIOS ausschalten lässt. Auch Sata-Controller mit ASMedia-Chip können Ärger machen, ebenso einige USB3-Chips von Via und Firewire-Chips von Ricoh. Einen sehr langen Thread darüber gibt es hier:
https://bugzilla.kernel.org/show_bug.cgi?id=42679

c) Grafikkarten

Ich habe bisher kein Schema erkennen können, an dem man "gute" und "schlechte" Grafikkarten unterscheiden kann. Theoretisch sollten sich alle Grafikkarten durchreichen lassen, praktisch wir das eh nur mit Karten von AMD oder Nvidia gemacht.
Wie oben schon erwähnt, versucht Nvidia einem hierbei Steine in den Weg zu legen. Neben einem besonders komplizierten Aufbau der Firmware sind das bisher nur angebliche "Bugs" in neueren Treibern, die einer Grafikkarte in einer VM den Betrieb verweigern. Bisher gibt es "Workarounds" für alle bekannten Probleme, aber das mag sich mit dem nächsten Treiber-Release schon wieder ändern. Solange

Nvidia bei dieser Politik bleibt, sind die Zukunftsaussichten ungewiss.

d) Neukauf

Wer eh schon seinen Rechner zuhause stehen hat, dem bleibt so oder so nichts anderes übrig als zu testen, ob es funktioniert. Ansonsten macht es Sinn, sich vorher schlau zu machen, welche Hardware denn bekanntermaßen IOMMU unterstützt. Wenn sonst nichts gegen AMD spricht, ist AMD eine ernste Überlegung wert. Zum einen sind die Systeme günstiger, zum anderen ist das oben angesprochene "VGA-Problem" bei AMD weniger schlimm. Der freie radeon-Treiber für die onboard-Grafik kommt wesentlich besser mit zwei aktiven VGA-Einheiten klar als der Intel-Treiber. Praktisch verwendet und getestet wird aber anscheinend mehr Intel.

Die zukünftige Entwicklung geht in Richtung "virtuelles EFI" statt "virtuelles BIOS". Dadurch, dass man die Grafikkarte im EFI-Modus startet, vermeidet man diese VGA-Probleme komplett. Daher ist es sinnvoll, bei Neuanschaffungen darauf zu achten, dass die Grafikkarte bereits über eine EFI-Firmware verfügt. Die Seite techpowerup.com listet bekannte Firmwares auf:

http://www.techpowerup.com/vgabios/

4) Die Hardware in Betrieb nehmen

Ziel dieses Abschnittes ist es, erstens die IOMMU zu aktivieren und zweitens dafür zu sorgen, dass die durchzureichende Grafikkarte weder vom BIOS noch vom Betriebssystem angefasst wird, sondern gleich mit dem vfio-Treiber belegt wird.

Dazu muss zuerst im BIOS alles, was nach "Virtualization", "vt-x", "vt-d" oder "AMD-Vi" klingt, eingeschaltet werden. Außerdem muss im BIOS eingestellt sein, dass die onboard-Grafik zuerst aktiviert wird - von der soll ja gebootet werden.

Danach muss per Kernel-Parameter die IOMMU aktiviert werden - die ist nämlich standardmäßig ausgeschaltet, weil es so viele problematische Hardware gibt. Für Intel-CPUs lautet der Parameter , für AMD-CPUs .

Gleichzeitig kann man schon mal die Kernel-Module, die die Grafikkarte belegen, auf die Blacklist setzen, damit sie nicht geladen werde. Dafür gibt es auch elegantere Methoden, wie das Einbinden des vfio-Treibers in die initramdisk, die bei mir allerdings allesamt Probleme machen oder nicht funktionieren. Die nötigen Kernel-Parameter trägt man am besten in die Datei /etc/default/grub ein, unter "GRUB_CMDLINE_LINUX_DEFAULT". Bei mir sieht die Zeile zum Beispiel so aus:

Code: Alles auswählen

```
GRUB_CMDLINE_LINUX_DEFAULT="modprobe.blacklist=radeon,snd_hda_intel intel_iommu=on intremap=no_x2apic_optout"
```

Ich hindere hier den radeon-Treiber daran, meine AMD-Grafikkarte zu belegen, und da die Karte auch eine HDMI-Soundkarte enthält, blockiere ich den snd_hda_intel-Treiber gleich mit. Dadurch bleibt meine Onboard-Soundkarte auch stumm ... das ändere ich später. Nach den Änderungen fehlt natürlich ein "update-grub".

Nun gilt es, den Rechner neu zu booten ... eventuell klappt das gar nicht, und man sieht Unmengen an Fehlermeldungen vorbeirauschen. Da hilft nur "Ausschalten" und nach der problematischen Hardware suchen. Eventuell hilft auch ein neuerer Kernel - aktuell ist Kernel 4.0 in "Unstable", der läuft bei mir sehr gut, nachdem ich auch den intel-xorg-Treiber aus Unstable installiert habe.

Klappt das booten, sollte man die gesamte Ausgabe von "dmesg" gründlich durchforsten (das hat man natürlich vorher schon mal gemacht) und nach neuen Fehlermeldungen suchen. So habe ich auch den Hinweis auf den Parameter "intremap=no_x2apic_optout" gefunden, den ich oben verwende.

Parameter, um weitere Problemfälle zu beheben, findet man eventuell hier:
https://www.kernel.org/doc/Documentatio ... meters.txt (nach "iommu" suchen. Und "ausprobieren". Ich versteh da auch nicht alles!)
Die Kernelmeldungen beim erfolgreichen Aktivieren der IOMMU ändern sich leider ständig, einen ersten Überblick verschafft oder . Ein simples "enabled" reicht bei Intel nicht, das ist nur die Software-Emulation, die funktioniert immer. Es sollten sich mehrere Zeilen finden, die das Stichwort "dmar" enthalten, und ein "Setting identity map" für einige PCIe-Geräte wäre auch gut.

Als nächstes schaut man sich die Ausgabe von "lspci -k" an. Die Grafikkarte und eventuell zugehörige HDMI-Soundkarte sollten keine Zeile "kernel driver in use" haben. Hierbei kann man sich auch gleich die PCI-Adressen der betreffenden Geräte merken - die stehen ganz am Anfang der Zeilen.

Danach ist die Ausgabe von "lspci -n" interessant. Wir brauchen die PCI-IDs der betreffenden Geräte. Die stehen ganz am Ende der Zeilen und sehen z.B. so aus "1002:aac0".

Über diese PCI-IDs können wir die Geräte jetzt an den vfio-Treiber binden. Das geschieht bei mir z.B. über folgende Zeilen:

Code: Alles auswählen

```
modprobe vfio
modprobe vfio_pci
echo 1002 6658 > /sys/bus/pci/drivers/vfio-pci/new_id
echo 1002 aac0 > /sys/bus/pci/drivers/vfio-pci/new_id
```

Ein "lspci -k" sollte jetzt anzeigen, dass die betreffenden Geräte vom "vfio-pci"-Treiber belegt sind. Wenn das nicht geklappt hat, hilft

wieder ein Blick ins "dmesg", um Fehlermeldungen zu finden.

Nun kann ich auch problemlos ein ausführen, um meine Onboard-Soundkarte in Betrieb zu nehmen.

Wenn diese Zeilen reibungslos klappen, sollten sie in ein Startup-Script, das vor dem Einloggen ausgeführt wird, zum Beispiel in die /etc/rc.local.

In diesem Script kann man auch gleich die kvm-Module laden:

Code: Alles auswählen

```
rmmod kvm-intel
rmmod kvm
modprobe kvm ignore_msrs=1
modprobe kvm-intel
```

(für AMD-CPUs nimmt man "kvm-amd" statt "kvm-intel")

Die MSRs sind Hardware-Register, die eigentlich von qemu emuliert werden sollten. Da es davon zig verschiedene gibt, emuliert qemu nicht alle. Bei Benutzung von nicht-emulierten MSRs gibt qemu eigentlich eine Fehlermeldung zurück, was die VM oder ein darin laufendes Programm üblicher Weise zum Absturz bringt. Der Parameter "ignore_msrs=1" sorgt dafür, dass qemu einfach "Null" zurückgibt, wenn so ein ununterstütztes Register ausgelesen wird. "Eigentlich" soll man ununterstützte MSRs an qemu melden, aber nach meiner Erfahrung guckt man als letztes ins "dmesg", wo solche Fehler gemeldet werden, wenn in der VM irgendwas nicht stimmt, und "Null" ist meistens eine gute Antwort. Wer trotzdem auf Probleme stößt, sollte "ignore_msrs=1" zuerst weglassen und schauen, was "dmesg" zu den Problemen sagt.

Übrigens: wenn PCI-Adressen nicht auf .0 enden, dann spricht man von "Multifunktionsgeräten". Die müssen fast immer am Stück durchgereicht werden, da sie eine vfio-Gruppe bilden. Die HDMI-Soundkarte einer Grafikkarte gleichzeitig unter Debian verwenden zu wollen, während die Grafikkarte selber an Windows weitergereicht wird, ist also eine dumme Idee. Die Anzahl dieser vfio-Gruppen sieht man unter /dev/vfio. Wenn eure Hardware IOMMU nicht gut genug umsetzt, dann landen in einer Gruppe zuviele Geräte und ihr habt ein Problem. Weiterführende Erklärungen zu diesen Gruppen finden sich hier:

https://www.kernel.org/doc/Documentation/vfio.txt

So, wir haben jetzt die Grafikkarte zum Durchreichen vorbereitet und können uns um die Virtualisierungssoftware kümmern.

5) virt-manager in Betrieb nehmen

Ein

Code: Alles auswählen

```
apt-get install libvirt-clients libvirt-daemon virt-manager qemu
```

sollte eigentlich reichen. (Das darf gerne noch mal jemand testen.) Danach findet man den virt-manager im Menü und kann ihn starten ... oder auch nicht, denn:

Wer "kvm" benutzen will, der sollte in der Gruppe "kvm" sein, und wer "libvirt" benutzen will, der sollte in der Gruppe "libvirt" sein. Nach diesen Änderungen ist ein logout/login fällig.

Ich mag jetzt nicht noch mal die Bedienung dieses GUI-Programmes genau beschreiben, zumal sich da nicht viel geändert hat - wer das lesen will, kann sich mein altes Tutorial angucken. Aber neue VMs lassen sich nur über den "Wizard" erzeugen - das ist das Symbol ganz links mit dem gelben Fleck. Die Bedienung ist eigentlich recht intuitiv, und es werden anhand des eingelegten Installationsmediums schon die richtigen Voreinstellungen getroffen.

Die Windows-DVD sollte im Laufwerk liegen, oder als ISO unter /var/lib/libvirt/images/ liegen.

Erst mal gilt es, ein virtuelles Windows zu installieren und zum Laufen zu kriegen. Das sollte gelingen. Mehrere Versuche sind erlaubt, und wer mit der virtuellen Hardware rumspielen möchte - das wäre der richtige Zeitpunkt. Wenn was nicht funktioniert, sollte der erste Blick ins "dmesg" gehen.

Virt-Manager erzeugt ein virtuelles Netzwerk "default". Gemeiner Weise wird das bei späteren Starts nicht automatisch gestartet, das muss man per Hand machen - mit - sonst gibt's ne Fehlermeldung.

Virt-Manager legt auch das Windows-Festplatten-Image unter /var/lib/libvirt/images/ ab. Wen die dicken Dateien unter /var stören, der kann mit "Edit->Connection Details->Storage->Add" einen neuen "Storage Pool" aufmachen und die Dateien dort ablegen.

Virt-Manager legt ein paar Geräte an, die die Virtualisierung beschleunigen sollen. Es ist empfehlenswert, Windows mit geeigneten Treibern dafür zu versorgen. Die findet man entweder hier:

http://www.spice-space.org/download.html als "spice-guest-tools-0.100.exe", oder hier:

https://fedoraproject.org/wiki/Windows_Virtio_Drivers als ISO oder Disketten Image.
Es sollten keine gelben Ausrufezeichen mehr im Windows-Gerätemanager zu sehen sein.

Das eigentliche Durchreichen der Grafikkarte ist dann recht einfach: Man öffnet das Fenster der VM, klickt auf die Glühbirne, klickt auf "Add Hardware", wählt "PCI Host Device" aus und erkennt seine Grafikkarte und eventuell HDMI-Soundkarte an der PCI-Adresse.

Das war's schon ... VM starten und Gerätemanager angucken!

Nun fehlen noch die Treiber für die Grafikkarte. Der AMD-Treiber-Installer ist mir wiederholt abgestürzt - macht nix - die Treiber sind in C:\AMD ausgepackt und können von dort mit dem Gerätemanager installiert werden. Mit Nvidia fehlt mir die eigene Erfahrung, aber man sollte sich eine möglichst alte Treiberversion suchen, um die oben erwähnten Probleme zu vermeiden. Auch hier gilt - wenn etwas nicht klappt, zuerst im dmesg nachgucken.

6) Manuell nachbessern

Etwas, was man auf kurz oder lang können möchte, das ist, die XML-Beschreibungen der VMs per Hand zu editieren. Der Befehl dazu lautet:

virsh edit <NameDerVM> (beim ersten Aufruf wird man nach seinem Lieblingseditor gefragt) Das Editieren per Hand ist tückisch ... was auch immer man da rein schreibt, virsh prüft es nachher nach seinen strengen Regeln, und beschwert sich entweder, wenn etwas nicht stimmt, oder nimmt auch klammheimlich Nachbesserungen vor, wie zum Beispiel das Anlegen eines Sata-Controllers, wenn man eine Sata-Festplatte hinzugefügt hat. Ein zweiter prüfender Blick nach dem Speichern kann nicht schaden. Überhaupt kann es nicht schaden, so eine XML-Datei ein wenig anzustarren, damit man sich ihren Aufbau grob einprägt.
Der Block, um ein (!) PCI-Gerät durchzureichen, sieht zum Beispiel so aus:

Code: Alles auswählen

```
<hostdev mode='subsystem' type='pci' managed='yes'>
   <source>
      <address domain='0x0000' bus='0x02' slot='0x00' function='0x0'/>
   </source>
   <address type='pci' domain='0x0000' bus='0x00' slot='0x08' function='0x0'/>
</hostdev>
```

Im source-Block steht die PCI-Adresse des durchzureichenden Gerätes.

In der zweiten address-Zeile steht, wo das Gerät in der VM landet. Der i440FX-Chipsatz hat hier nur einen Bus "bus='0x00'", nämlich den PCI-Bus. Interessant ist aber der slot='0x08'. Mit diesem Slot kann man herumexperimentieren, wenn Windows einen Code 12 meldet, und versuchen, dort eine andere Nummer einzutragen, um das Gerät auf einen anderen IRQ zu legen (den Erfolg sieht man dann im Windows-Gerätemanager).
Bei einem Code 43 hilft hingegen nach meiner Erfahrung eher das Entfernen eines der störenden Geräte, oder ein besserer Windows-Treiber.

Weiter oben sieht man den Block

Code: Alles auswählen

```
<pm>
  <suspend-to-mem enabled='no'/>
  <suspend-to-disk enabled='no'/>
</pm>
```

Das "suspend-to-disk" kann man nach meiner Erfahrung ruhig auf "yes" stellen. Dann lässt sich Windows schlafen schicken, und danach kann man auch sein Debian in den Suspend schicken.

Virt-Manager erlaubt es nicht mehr, ein SDL-Fenster zu erzeugen. Libvirt aber schon. Dazu muss man den "<graphics type" zu "<graphics type='sdl' ..." ändern.

7) Tipps und Tricks

a) Nvidia-Karten zum Laufen kriegen

Das Problem steckt im Treiber von Nvidia. Der versucht, das kvm-Modul zu erkennen, und meldet dann verwirrender Weise einen "Code 43".

Das tut er ab Version "337.88" - die einfachste Abhilfe ist also ein älterer Treiber.

Sonst muss kvm vor dem Treiber versteckt werden. Das geschieht mit <hidden state='on'/>:

Code: Alles auswählen

```
<domain type='kvm'...>
```

```
  <features>
    <kvm>
      <hidden state='on'/>
    </kvm>
  </features>
  ...
</domain>
```

Weiterhin dürfen folgende zwei Code-Blöcke **nicht** verwendet werden:
Code: Alles auswählen

```
<hyperv>
      <relaxed state='on'/>
      <vapic state='on'/>
      <spinlocks state='on' retries='8191'/>
</hyperv>
```

und
Code: Alles auswählen

```
<clock offset='localtime'>
    <timer name='hypervclock' present='yes'/>
  </clock>
```

Die werden aber (zumindest bei mir) eh nicht angelegt.

Da das für mich eine reine Trockenübung ist, gebe ich mal die Quellen mit an:

http://vfio.blogspot.de/2014/08/vfiovga-faq.html
http://vfio.blogspot.de/2014/08/upstrea ... -2014.html

b) "Windows zerstören" für Fortgeschrittene

Ich habe es wie gesagt geschafft, mir Konfigurationen zu basteln, die durch das Einspielen der Windows-Updates kaputt gegangen sind.

Ebenso habe ich es geschafft, Windows in eine endlose Reparaturschleife zu bringen. Dagegen half auch kein Wiederherstellen der vorherigen Konfiguration. Ich habe es sogar geschafft, meine AMD-Grafikkarte in einen Zustand zu bringen, bei der sie am PCI-Bus nicht mehr angezeigt wurde. Dagegen half nicht mal ein Reboot. Es half nur ein Herunterfahren, Netzkabel ziehen, eine Minute warten.

Daraus schließe ich, dass wer auf Probleme stößt und mehrere Konfigurationen durchprobieren will und dabei wirklich pedantisch vorgehen will, bei jeder Änderung einen Kaltstart machen muss und sein Windows neu installieren muss. Umgekehrt bedeutet das aber auch - wer ein "funktioniert nicht" vermelden will, der sollte zumindest so gründlich sein, um das guten Gewissens behaupten zu können.

Wie auch immer ... wenn Windows erst mal Probleme sucht, dann hilft nach meiner Erfahrung gar nichts mehr. Eine Neuinstallation (von Windows) ist fällig.

c) Qemu-Optionen direkt verwenden

Da Libvirt und vorallem Virt-Manager etwas eingeschränkt sind und nicht sämtliche Funktionen von qemu unterstützen, ist es manchmal nötig, qemu-Optionen direkt zu verwenden.

Man sieht öfter Beispiele, in denen Qemu-Optionen direkt in die XML-Dateien von Libvirt eingebettet werden. Das passiert innerhalb von <qemu:commandline>-Tags. Damit zieht man einen Rattenschwanz von Problemen hinterher. Vorallem beim pci-passthrough kann Libvirt dann die Verwaltung der vfio-Gruppen nicht mehr selber durchführen, und muss daher als "root" betrieben werden, wodurch einige andere Sachen, wie ein SDL-Fenster oder die Sound-Ausgabe nicht mehr funktionieren, von den Sicherheitsrisiken mal ganz abgesehen.

Eleganter ist es, ein "Wrapper-Script" zu verwenden. Das wird z.B. gespeichert unter

/usr/sbin/qemu-system-x86_64-wrapper und enthält als Rumpf:
Code: Alles auswählen

```
#!/bin/sh
    exec /usr/sbin/qemu-system-x86_64 `echo "$@" | sed -r 's|<zu ersetzender Parameter>|<gewünschter Parameter|'`
```

Das echo "$@" gibt die von Libvirt erzeugte Qemu-Optionsliste aus, und per sed werden dann Korrekturen daran durchgeführt.
In der XML-Datei muss dann nur

<emulator>/usr/sbin/qemu-system-x86_64</emulator> geändert werden zu <emulator>/usr/sbin/qemu-system-x86_64-

wrapper</emulator> und schon wird das Wrapper-Script verwendet.
Mit kann man bei laufender VM die aktuell verwendete qemu-Befehlszeile herausfinden. Das ist praktisch, um das sed-Kommando erst mal als Trockenübung auszuprobieren.

d) OVMF verwenden

Qemu bringt ein virtuelles BIOS mit, um damit VMs starten zu können - das "Sea-Bios". Da der Trend immer mehr zu UEFI geht, ist ein virtuelles UEFI in Entwicklung - das nennt sich OVMF.

Das ist spannend zum Ausprobieren von UEFI-Installationen, außerdem schwört BLACKDIAMOND in seinem Thread hier auf OVMF als einzige Methode, mit der vga-passthrough bisher bei ihm geklappt hat:

viewtopic.php?f=12&t=154032
Jessie bringt ein OVMF-Image mit, das steckt im Paket "ovmf" und findet sich nach dem Installieren unter /usr/share/ovmf/OVMF.fd. Von dort sollte es zur Verwendung irgendwohin kopiert werden.

Nun unterstützt der Virt-Manager noch kein OVMF, libvirt aber schon, und das Einbinden des OVMF-Images ist sehr einfach:

http://vfio.blogspot.de/2014/09/libvirt ... -ovmf.html
Das klappt bei mir auf Anhieb, das Booten aber nicht - dazu muss ich erst auf "F8" rumhämmern, um den Boot Manager aufzurufen. Eine Test-Installation von Win7-64Bit scheiterte dann an einem schwarzen Bildschirm. Das liegt am Alter des OVMF-Images ... die Version von Jessie scheint von 2013 (!) zu sein.

In sid/testing findet sich aktuell ein OVMF von 2015. Die Installation klappt problemlos, es gibt keine Abhängigkeiten. Damit läuft auch die Installation von Win7 problemlos durch, und vga-passthrough geht auch - das ging bei mir aber auch schon vorher. Eine Grafikkarte mit EFI-Firmware habe ich nicht zur Verfügung.

Achtung: ein EFI muss auch Variablen speichern können - im Gegensatz zum BIOS, das auch als "read-only" funktioniert. Es ist also für jede VM ein eigenes OVMF-Image nötig, in dem die VM ihre Einstellungen hinterlegen kann.

Achtung: es gibt "reines EFI" (pure EFI) und "EFI mit CSM Modulen". Eine Grafikkarte, die keine EFI-Firmware hat, wird auf einem "reinen EFI" eventuell nicht starten. Eine Grafikkarte, die eine EFI- und eine VGA-Firmware hat, kann mit einem "reinen EFI" dazu gezwungen werden, im EFI-Modus zu starten (das ist der eigentliche Sinn der Sache beim vga-passthrough). Bei einer Grafikkarte mit

EFI- und VGA-Firmware ist es bei einem "EFI mit CSM Modulen" Glückssache, ob die Karte im VGA- oder im EFI-Modus startet.

e) Q35 verwenden

Ich habe durch Q35 bisher keine Vorteile feststellen können, daher mag ich es nicht empfehlen. Einige Leute schwören aber darauf, dass vga-passthrough bei ihnen nur mit Q35 funktioniert, daher sollte es erwähnt werden.

Der Virt-Manager von Jessie unterstützt den Q35-Chipsatz von qemu noch nicht. Libvirt aber schon.

Da ich eine faule Sau bin und nicht so viel tippen mag, habe ich mich nach einem neueren Virt-Manager umgeguckt. In Jessie steckt die Version 1.0 und für Q35 ist mindestens die Version 1.1 nötig. Die ist in Debian bisher nicht zu finden. Nötig sind aktuellere Pakete von "virt-manager" und "virtinst". Die habe ich dann hier aufgetrieben:

http://www.ubuntuupdates.org/pm/virtinst
http://www.ubuntuupdates.org/pm/virt-manager
Die Installation war etwas hakelig, da einige gir1.2-* Pakete vorausgesetzt werden - aus den Jessie-Repositories natürlich. Als nächstes muss virtinst installiert werden, da virt-manager davon abhängt. Das ist für mich nur eine Notlösung - Debian-Pakete wären mir natürlich lieber.

Die Unterstützung von Q35 ist auch noch etwas hakelig. Q35 kann nur während der Erzeugung einer neuen VM ausgewählt werden, nicht mehr danach, und auch nur, wenn man "customize before install" auswählt. Dabei legt Virt-Manager ein IDE-CDRom-Laufwerk an, was so von Q35 nicht mehr unterstützt wird - das muss manuell auf "Sata" gestellt werden, sonst kriegt man eine verwirrende Fehlermeldung.

Aus Gründen der Kompatibilität nutzt Libvirt bisher die PCIe-Möglichkeiten von Q35 **nicht** für pci-passthrough! Es wird eine PCIe->PCI-Brücke erstellt, und die meisten Geräte landen an diesem PCI-Bus. Ein vga-passthrough gelang mir damit durch "Zurechtklicken" **nicht**! Es muss also mit direkten qemu-Optionen gearbeitet werden.

Bei mir reichte es, die vfio-pci-Optionen, die Libvirt erzeugt, durch folgende zu ersetzen:

Code: Alles auswählen

```
-device vfio-pci,host=02:00.0,bus=pcie.0,addr=09.0,multifunction=on
-device vfio-pci,host=02:00.1,bus=pcie.0,addr=09.1
```

(beides natürlich auf einer Zeile)

Der "host="-Parameter enthält hier die PCI-Adressen meiner Grafikkarte und dessen HDMI-Soundkarte.

Der "addr="-Parameter bestimmt, an welcher Adresse in der VM die Geräte landen. Mit dem "09" sollte man experimentieren, falls ein "Code 12" auftritt. Das "09" ist übrigens eine Hex-Zahl - die kann bis z.B. "0d" oder "1c" hochgezählt werden.
Die Kombination aus OVMF und Q35 habe ich übrigens nicht zum Laufen bekommen. Das UEFI findet weder CD-Rom noch Festplatte zum Booten.

Sollte das hier nicht mal ins Wiki?

Vermutlich nicht - dazu ist es zu lang, zu allgemein, zu umständlich, enthält vermutlich Fehler und Ungenauigkeiten und veraltet zu schnell. Wahrscheinlich müsste es aufgespalten werden in einzelne Abschnitte, was es wiederum auseinanderreißen würde. Und "alleine" binde ich mir so ein Projekt und die Pflege desselben garantiert nicht ans Bein.

Aber wer mag, der darf diesen Text kopieren, weitergeben, verändern, verkaufen, oder sogar ins Wiki übernehmen.

Erst mal halte ich aber Feedback für sinnvoll ... also wer mag und kann, kann die Sache gerne ausprobieren!