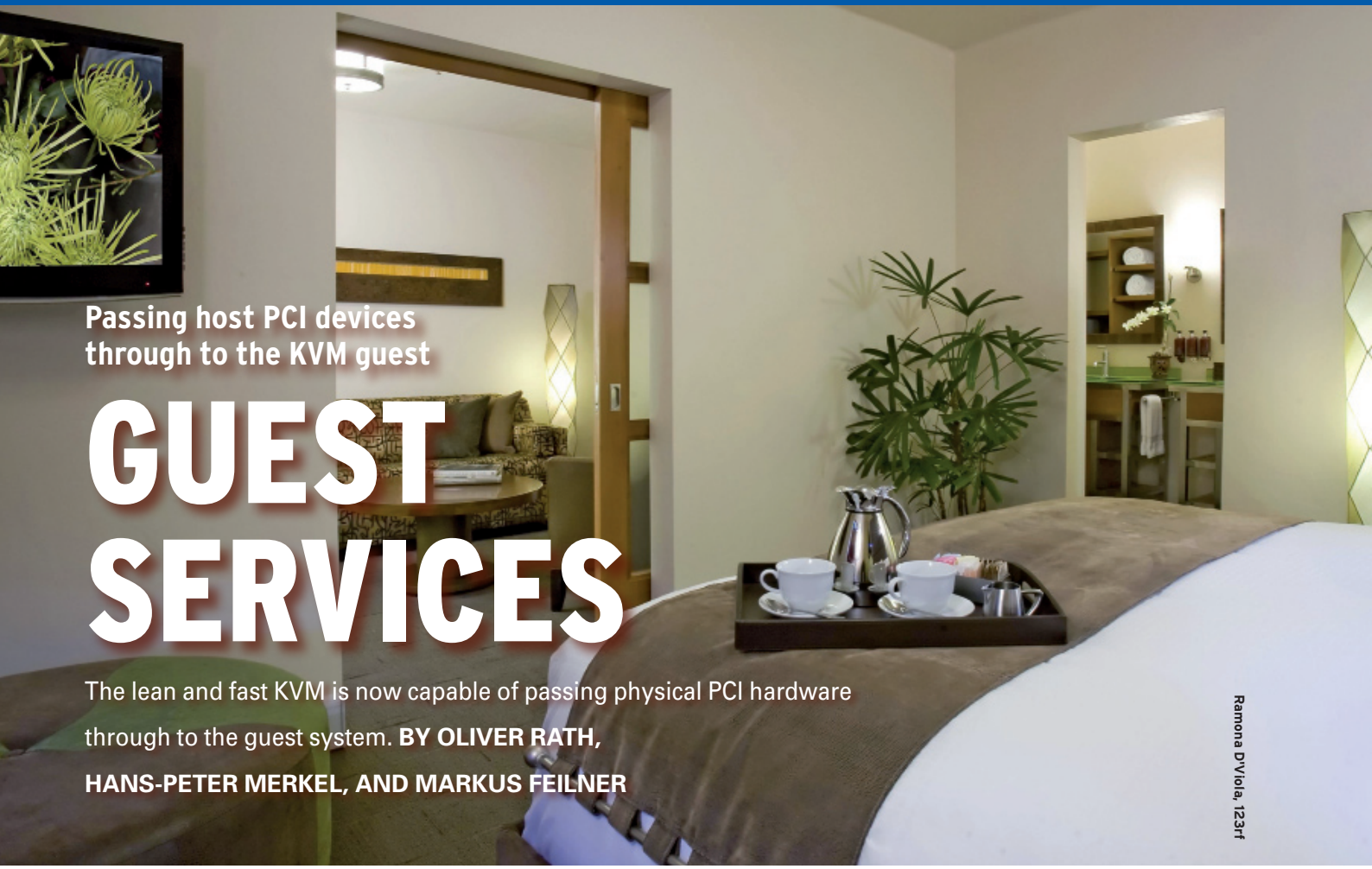

Ramona D'Viola, 123rf

Passing host PCI devices through to the KVM guest

GUEST SERVICES

The lean and fast KVM is now capable of passing physical PCI hardware through to the guest system. **BY OLIVER RATH, HANS-PETER MERKEL, AND MARKUS FEILNER**

The popular KVM virtualization system found with many Linux systems now provides hardware-based "passthrough" access to real PCI devices. PCI passthrough can provide better performance than conventional hardware emulation techniques, and in some cases, it might be the best and most practical solution for connecting to an unusual PCI device that, for whatever reason, isn't accessible from the guest through emulation. To achieve the best possible support for PCI passthrough, you are better off with the latest KVM kernel modules – or the latest kernel – but at least 2.6.26, preferably with the Userspace-Tools 0.12.2 in place [1].

The restriction of only running on Intel processors with Vanderpool support, or on AMD's counterpart, Pacifica, is not an issue. Most CPUs, with the exception of some Atoms, Celerons, and OEM series for low-budget laptops, come with these extensions built in. AMD includes them with all recent CPUs since the introduction of the dual cores (with the exception of Sempron).

As a general rule, Linux can only pass hardware that the host does not use itself through to a virtualized instance. For this to happen, you first need to unload the modules, or prevent loading by means of a blacklist. As is typically the case on Gentoo, it is a good idea not to bind the modules with your own kernel.

Switch off IRQ Sharing

Disabling the shared interrupt functionality is very important – and not just for kernel builders. Currently, KVM can only pass through PCI devices with their own interrupts. Passthrough fails if multiple devices share an interrupt. To identify duplicate use, it is a good idea to scan

A Little History

KVM began as a fork of QEMU, and the development of KVM and QEMU continued in parallel for some time.

The developers' plan was to get people interested in QEMU because of the GPL license and then earn money through commercial licensing of the KQEMU module, which accelerated the emulator to almost five times the speed. Unfortunately, the introduction of Vanderpool hardware with hardware virtualization made this obsolete, or at least that's how one of the developers with the company behind KVM, Avi Kivity of Qumranet, puts it in his blog [2].

Debian Matters

The Debian Lenny repositories still return the obsolete KVM 77 version if you query the version by issuing *kvm -version*. (The upcoming Debian Squeeze, as well as Ubuntu 9.10 and other Debian derivatives, give you v0.11.1.)

The packages for the i386 or AMD64 are available for downloading from the Debian server. To do so, become root and give the *dpkg -i* command to install. If you installed an older version previously, you should not experience any dependency problems.

qemu-kvm --help should give you a success message (Listing 4). Now you can start your first lending action on Debian.

THE AUTHORS

Oliver Rath is a co-founder of LPI Germany. His work focuses on virtualization with KVM, green IT, integration of VoIP (Asterisk), automation of transactions in public offices, migration to free software, training, and consulting.

Hans-Peter Merkel has been an active member of the Open Source community for many years, focusing on data forensics. He also trains law enforcement officers in Germany and Tanzania, and he is the founder and chair of FreiOSS and Linux4Afrika.

the interrupts currently in use with *lspi -vv | grep IRQ* (Listing 1), then compare the output with the device you want to pass through (Listing 2). In this example, passthrough will not work because interrupt 11 is shared by three devices.

Listing 3 is an example with a four-port PCIe ISDN card by Cologne Chip Designs. Its interrupt (19) is only used once by the ISDN card. Admins of Ubuntu or Debian systems can enter the modules they do not need in */etc/modprobe.d/blacklist.conf*:

```
blacklist mISDN_dsp
blacklist hfcmulti
blacklist mISDN_core
```

The example shows the modules for the active HFC4S ISDN card; these are drivers for the new *mISDNv2* subsystem, which was introduced with kernel 2.6.28 and is highly recommended for HFC ISDN cards. If you are using the Asterisk telephony tool, you also need the channel driver *lcr_chan* from the Linux Call Router project [3].

Finding the PCI ID

Before passing a PCI or PCIe card through to a virtualized operating system, you need the PCI ID, which you can find with the *lspci* command

```
[...]
00:11.0 Network controller:
   AVM GmbH B1 ISDN
```

and then pass in to KVM as a start-up parameter.

ISDN for the Guest

Suppose you want to set up the guest system to use the Capisuite fax answering machine tool connected through a Fritz ISDN card on the host system. The first problem is that the only packages available are RPMs for SUSE, and modifying them is hard work. The newer the Debian kernel, the more difficult it is to stop the compiler from complaining. The example that follows shows a Capisuite fax answering machine solution on Debian Etch, in which the modules and compiler do not cause a problem because the system still uses kernel 2.6.18. KVM launches the guest in the background and assigns a separate IP address to it for SSH access.

After installing a standard system, the guest still needs the *build-essential*, *rpm*, and *capiutils* packages, along with the headers. The guest system also needs Fritz card drivers. Because the host typically uses these drivers exclusively, with the ancient Hisax ISDN module, you will need a blacklist. The following two lines in */etc/modprobe.d/blacklist.conf* are all it takes:

```
blacklist hisax_fcpcipnp
blacklist hisax
```

The host will now ignore these drivers, and the guest will not detect the new hardware until KVM tells it to. The guest originally thinks that Hisax is the right module. To prevent this from happening, you need to create an */etc/modprobe.d/blacklist-capisuite* file on Etch with content identical to the entries shown in *blacklist.conf* above.

On launching *kvm*, you need to pass in the device ID on the command line. The guest system should now load the module:

```
gast # lsmod | grep fcpci
fcpci 592768 1
kernelcapi 43680 2 capi,fcpci
```

Capiinfo should now tell you that a PCI card is available on the guest (Listing 5). There is nothing to stop you from configuring Capisuite as a fax server and answering machine.

Listing 1: lspi --vv | grep IRQ

```
01 Interrupt: pin D routed to IRQ 10
02 Interrupt: pin D routed to IRQ 12
03 Interrupt: pin D routed to IRQ 12
04 Interrupt: pin ? routed to IRQ 9
05 Interrupt: pin A routed to IRQ 11
06 Interrupt: pin A routed to IRQ 12
07 Interrupt: pin A routed to IRQ 10
08 Interrupt: pin A routed to IRQ 11
09 Interrupt: pin A routed to IRQ 11
```

Listing 2: lspci -vv | less

```
01 [...]
02 00:11.0 Network controller: AVM GmbH B1 ISDN
03 Control: I/O+ Mem+ BusMaster+ SpecCycle- MemWINV-
   VGASnoop- ParErr- Stepping- SERR- FastB2B- DisINTx-
04 Status: Cap- 66MHz- UDF- FastB2B- ParErr- DEVSEL=fast
   >TAbort- <TAbort- <MAbort- >SERR- <PERR- INTx-
05 Latency: 32
06 Interrupt: pin A routed to IRQ 11
07 Region 0: I/O ports at d800 [size=64]
08 Region 1: I/O ports at dc00 [size=32]
09 Kernel driver in use: b1pci
10 Kernel modules: b1pci
 11 Capabilities: [40] Power Management version 3
112 [...]
```

Listing 3: lspci -vv

```
01 08:04.0 ISDN controller: Cologne Chip Designs GmbH ISDN
   network Controller [HFC-4S] (rev 01)
02 Subsystem: Cologne Chip Designs GmbH Device b752
03 Control: I/O- Mem+ BusMaster- SpecCycle- MemWINV-
   VGASnoop- ParErr- Stepping- SERR- FastB2B- DisINTx-
04 Status: Cap+ 66MHz- UDF- FastB2B- ParErr- DEVSEL=medium
   >TAbort- <TAbort- <MAbort- >SERR- <PERR- INTx-
05 Interrupt: pin A routed to IRQ 19
06 Region 0: I/O ports at 5000 [disabled] [size=8]
07 Region 1: Memory at c6000000 (32-bit, non-prefetchable)
   [size=4K]
08 Capabilities: [40] Power Management version 2
09 Flags: PMEClk- DSI+ D1+ D2+ AuxCurrent=0mA
   PME(D0+,D1+,D2+,D3hot+,D3cold-)
10 Status: D0 NoSoftRst- PME-Enable- DSel=0 DScale=0 PME-
11 Kernel driver in use: hfc_multi
12 Kernel modules: hfcmulti
```

Listing 4: qemu-kvm --help

```
01 QEMU PC emulator version 0.10.50 (qemu-kvm-devel-88),
   Copyright (c) 2003-2008 Fabrice Bellard
02 usage: qemu [options] [disk_image]
03 [...]
04 -pcidevice host=bus:dev.func[,dma=none][,name=string]
   expose a PCI device to the guest OS.
05 dma=none: don't perform any dma translations (default is
   to use an iommu) 'string' is used in log output.
06 [...]
```

Suppose you want to install a DVB-T card on the guest system in addition to the ISDN card. The DVB-T card will be used as a digital video recorder with VDR. It makes sense to use a separate disk to store the huge amounts of data the recordings create.

Just as in the previous examples, you need to prevent the host from accessing the DVB card and enter the drivers *dvb_ttpci*, *stv0299*, *saa7146_vv*, and *saa7146* from Listing 6 in your */etc/modprobe/blacklist.conf*. You will need more than the kernel modules for most DVB cards; they additionally expect firmware in */lib/firmware*. Finally, KVM needs the PCI device ID for the DVB card. In this example, *lspci* lists a Philips SAA7146 chip as *00:06.0*.

PCI Hotplug in KVM

At run time, you can force the guest to use more PCI devices. The less well known QEMU Monitor console, launched with the shortcut Ctrl + Alt + 2, helps you add devices. The classic white-on-black background terminal gives users a number of practical commands. Detailed documentation is available, and if you type *help* at the prompt, you will find information on the syntax and the supported commands. *info pci* lists all known PCI devices in the virtual instance, and *pci_add* lets you add a device, such as another Ethernet card:

```
(qemu) pci_add auto nic model=e1000
OK domain 0 bus 0 slot 9 function 0
(qemu)
```

In a style similar to the syntax used in KVM, you can stipulate *host =* to define the host PCI ID. For this to work properly, the *acpiphp* and *pci_hotplug* kernel modules must be loaded on the guest system. In this case, *dmesg* will display detailed information on the new PCI devices, and the *lspci* list will just keep growing. However, there is more to it than that: You can use the Qemu Monitor console to add or remove drives, as well as USB and storage devices, more or less at will. When you are done, pressing Ctrl + Alt + 1 will return you to the familiar KVM window.

Hardware en Masse

The guest now has a great selection of hardware and exclusive access to the physical PCI devices (Listing 7). A feature that insiders and forensics specialists have always appreciated is thus becoming more and more interesting in other fields. Administrators can run custom solutions on protected systems; geeks can run video recorders in the background. And if this is not enough, you can wait for Via's Nano processor technology and dream of energy-saving embedded systems that run Windows and Linux in parallel. ■

Listing 6: lsmod | grep dvb

```
01 dvb_ttpci 104576 18
02 dvb_core 99120 2 stv0299,dvb_ttpci
03 saa7146_vv 49920 1 dvb_ttpci
04 saa7146 19160 2 dvb_ttpci,saa7146_
   vv
05 ttpci_eeprom 2672 1 dvb_ttpci
06 i2c_core 26736 7
   nvidia,stv0299,ves1x93,
   dvb_ttpci,videodev,ttpci_eeprom,
   i2c_piix4
```

Listing 7: lspci on the Guest

```
01 00:00.0 Host bridge: Intel Corporation 440FX - 82441FX PMC [Natoma] (rev 02)
02 00:01.0 ISA bridge: Intel Corporation 82371SB PIIX3 ISA [Natoma/Triton II]
03 00:01.1 IDE interface: Intel Corporation 82371SB PIIX3 IDE [Natoma/Triton II]
04 00:01.3 Bridge: Intel Corporation 82371AB/EB/MB PIIX4 ACPI (rev 03)
05 00:02.0 VGA compatible controller: Cirrus Logic GD 5446
06 00:03.0 Ethernet controller: Realtek Semiconductor Co., Ltd.
   RTL-8139/8139C/8139C+ (rev 20)
07 00:04.0 RAM memory: Unknown device 1af4:1002
08 00:05.0 Network controller: AVM Audiovisuelles MKTG & Computer System GmbH A1
   ISDN [Fritz] (rev 02)
09 00:06.0 Multimedia controller: Philips Semiconductors SAA7146 (rev 01)
```

Memory

The virtualized Linux derivative still needs memory. If you do not specify the *-m* parameter, KVM will assume a default RAM size of 128MB, which is definitely too small for Windows XP and probably far too small for most Linux distributions. The Fedora installer and YaST both run into trouble if you don't give them at least 256MB.

Users with virtualization servers that suffer from memory overloading can look forward to Kernel SamePage Merging (KSM) [4], which continuously scans occupied RAM pages and just keeps a single instance of each page in memory if they are identical. Red Hat, the new owners of Qumranet, and Fedora are continuing to develop this actively.

Listing 5: capiinfo on the Guest

```
01 Number of Controllers : 1
02 Controller 1:
03 Manufacturer: AVM GmbH
04 CAPI Version: 2.0
05 Manufacturer Version: 3.11-07
   (49.23)
06 Serial Number: 1000001
07 BChannels: 2
08 Global Options: 0x00000039
09 internal controller supported
10 DTMF supported
11 Supplementary Services supported
12 channel allocation supported
   (leased lines)
```

In the Background

You do not need a shell for the virtual system in production operations. The guest can run as a completely headless background process, wherein you use SSH for management tasks:

```
# qemu-kvm -m 1024 -net nic,
  vlan=0,macaddr=U
00:80:ad:11:11:11 -net tap
  -pcidevice host=U
05:06.0 -nographic
  -daemonize etch.img
```

To be able to assign a static IP address to the virtual system using DHCP, you need to pass in a virtual MAC address when starting the system and reserve the address for this system on the DHCP server.

INFO

[1] Kernel Virtual Machine: *http://www.linux-kvm.org*

[2] Avi Kivity's KVM-Blog: *http://avikivity.blogspot.com*

[3] Linux-Call-Router-Project: *http://www.linux-call-router.de*

[4] Kernel SamePage Merging: *http://fedoraproject.org/wiki/Features/KSM*