Virtualisierung mit KVM

Vortrag
anlässlich des
Stammtisches der
Erdinger Linux User Group
am
22. April 2009

von
Oliver Rath

<Werbung>

- Gründung von Rath EDV 1993
- Virtualisierungslösungen seit 2003
- Schwerpunkt OpenSource Migration und Green IT
- Erfolgreiche Kombination von GreenIT und Virtualisierung seit 2008 auf KVM-Basis

</Werbung>

Übersicht

- Begriffsklärung
- Motivation
- Historie
- KVM in der Theorie
- KVM in der Praxis
- Aktueller Stand
- Ausblick

Virtualisierung in der Literatur

Stanislaw Lem's „Prof. Corcorans Welt“
aus den Sternetagebüchern von Ion Tichy (1971):

„Unsere Hirne - geben Sie acht! - sind sozusagen an die äußere Welt angeschlossen, vermittels der Sinnesrezeptoren: der Augen, der Ohren, der Nase, der Haut und so weiter.
[…]
Woher wissen wir, was real ist?“

Überblick Virtualisierungslösungen

- Bochs
- BSD-Jails
- Crossover (€)
- DosEmu
- DosBox
- Frodo (C64)
- Hercules (Esa/390, zSeries)
- KVM
- LGuest
- OpenVZ
- PVM
- Qemu
- SkyEye (ARM)
- SpektEmu (ZX Spektrum)
- Spim (MIPS)
- Virtual Server (Microsoft)
- Virtualbox
- Virtuozzo
- VMWare (€)
- Vserver
- Wine
- Win4Lin
- X48 (HP 48)
- Xen

Begriffsklärung

- Was bedeutet Virtualisierung
- Spielarten
  - Vollvirtualisierung (Bochs)
  - Paravirtualisierung (vserver, openvz, virtuozzo, Xen)
  - Gerätevirtualisierung (KVM, XEN)

Motivation für Intel/AMD

- Hardware stark genug für Virtualisierung
- Virtualisierung für den Massenmarkt
- Parallelbetrieb von unmodifizierten Betriebssystemen
- **Sichere Implementierung von TPM (!)**

Pros und Contras von KVM

+

- Sehr schnell, da nur gerätevirtualisierend
- **Vollständig GPLv2**
- Nutzbar für x86-Architektur
- Linux als Hypervisor
- VirtIO-Architektur

–

- Nur auf x86-Architektur schnell
- Noch sehr jung (<4J)
- Wenig graphische Administration bisher
- Linux als Hypervisor
- (Noch) keine Hot-Migration
- (Noch) kein VirtIO-Blockdevice für MS-Windows
- Bei 32Bit-Host max 2GB/Gastsystem RAM

Historie: Hardware

- Intel
  - VT kommt 11/2005
  - Erster Prozessor mit VT: Pentium4-6x2
  - VT-Unterstützung extrem uneinheitlich
  - /proc/cpuinfo: Flag „vmx“
- AMD
  - Pacifica kommt 06/2006
  - **alle** Modelle mit DualCore und besser + Turion
  - /proc/cpuinfo: Flag „svm“

Historie: Software KVM

- Fork von Qemu 0.9.1
- Beinhaltet div. GPL-Software
  - OpenBios (seit 0.8.2, vorher „Proll“)
  - Bochs
- Refork nach Qemu mit der Version 0.10.1
- Keine einheitliche Namenskonvention:
  - kvm{,-img,-nbd} (Ubuntu, Gentoo) oder
  - qemu{,-img,-nbd} (Suse?)

Theorie: Intel Vanderpool (VT)

- Intel erweitert den Befehlssatz um 10 Befehle und den sog. Non-root-mode und den
- VMCS (Virtual Machine Control Structure), einen 4KByte-Block.
- Diese erweitern das Speichermapping von Ring 0 und erweitern sie die Behandlung des NMI
- Das eigentlich Programm läuft **unmodifiziert** auf der Intel-Hardware

Theorie: AMD Pacifica (AMD-V)

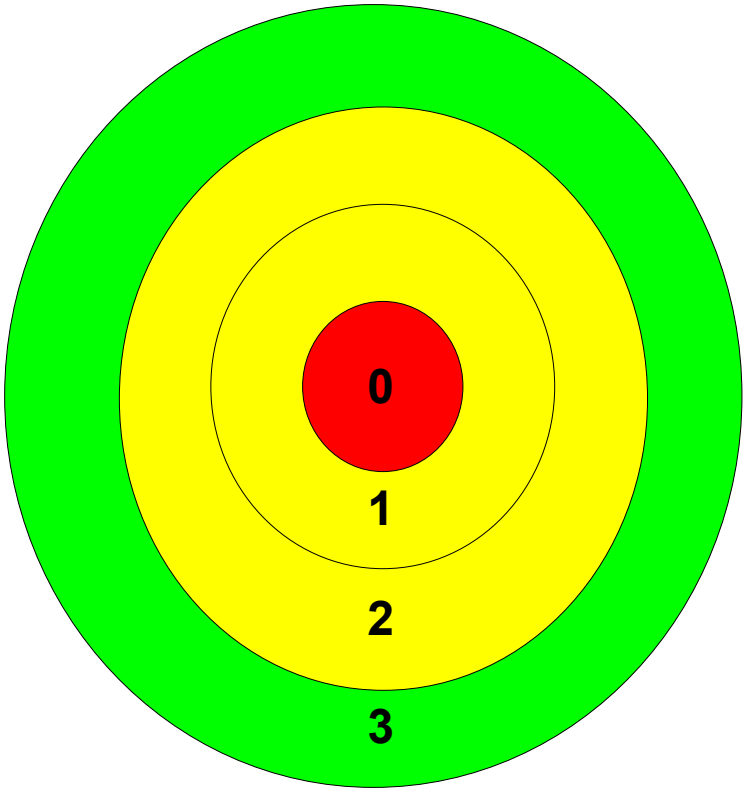
- AMD erweitert den Befehlssatz um 9 Befehle und den
- VMCB („Virtual Mode Control Block“)
- Neuer „Paged Real Mode“
- Auch hier unmodifizerte Ausführung möglich
- Zusätzlich zu Intel:
  - „NPT“ (Nested Page Tables)
  - Speichercontroller auf CPU integriert

Theorie: PowerPC5

- IBM virtualisiert seit Jahren
- Virtualisierung auch bei Netzwerkkarten, IO-Controllern etc.
- Hypervisor ist Firmware
- Die Software ist **proprietär**, wenig Informationen
- **KVM für PowerPC5 ist „work in progress“**
- Cell-Prozessor (Playstation III) basiert auf PowerPC5 ...

Theorie: Prozessor ohne VT

- Bisher:
  - Ausführungsprevile-gien 4-schichtig
  - Schicht 0: BS-Kern
  - Schicht 3: User
  - Schicht 1 & 2 unbenutzt

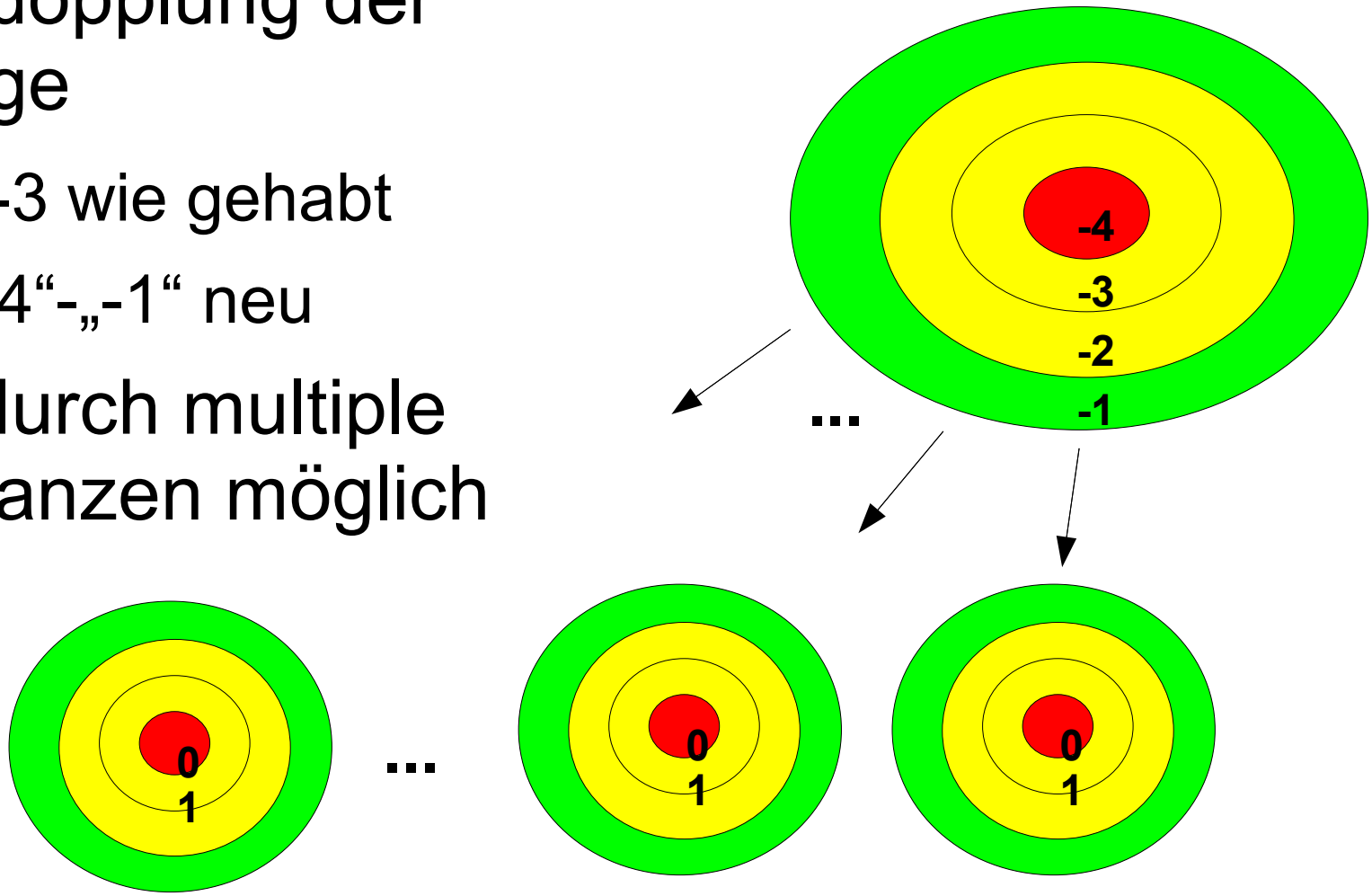


Betriebsystem

Anwendungen

Theorie: Prozessor mit VT

- Verdopplung der Ringe
  - 0-3 wie gehabt
  - „-4“-„-1“ neu
- Dadurch multiple Instanzen möglich



Praxis: Kernel

- KVM besteht aus zwei Kernelmodulen
  - Kvm (generisch)
  - Kvm-{intel,amd)
- Für besondere Netzwerktopologien: tap/tun
- KVM-Optimierungen für Guest-Betrieb möglich, aber nicht notwendig

Praxis: VirtIO-Schicht

- KVM kann seit 2.6.23 die VirtIO-Schicht im Kernel nutzen
- Folgende virtuellen Devices stehen im Moment zur Verfügung:

- Netzwerkdevice
- Blockdevice
- PCI-Device
- Console
- Balloon-Device (RAM)
- Zufallszahlengenerator

Praxis: Emulierte Hardware

- 8 Netzwerkkarten: ne2k_pci, i82551, i82557b, i82559er, rtl8139, e1000, pcnet, virtio
- Für x86-Architektur 11 Varianten: Qemu64, phenom, core2duo,qemu32, coreduo, 486, pentium{,2,3}, athlon, n270
- Maschinen: pc, isapc (ohne PCI)
- Sound: pcspk, sb16, ac97, es1379
- Grafik: cirrus, vga, std, vmware,
- Motherboard: i440FX host PCI bridge und PIIX3 PCI to ISA bridge
- Maus, Floppy, cdrom, Festplatte (ide, scsi, virtio)

Praxis: Grafik Umleitung

- Ausgabe auf mehrere Arten möglich
  - Per SDL
  - Per curses (bei Textconsole)
  - Per VNC
  - Direkt an PCI-Grafikkarte durchgereicht (nur bei unallocierten Grafikkarten, IRQ-Problematik noch nicht sauber gelöst)
  - Gar nicht (-nographic)

Praxis: Sonstige Hardware

- USB-Geräte können dynamisch (!) durchgeschleift werden
- Bluetooth ebenso
- Seit kvm-84 (14.1.2009) auch PCI-Geräte (noch nicht dokumentiert)
- Serielle Schnittstelle nach außen durchreichbar (wichtig für embedded!)

Praxis: Massenspeicher einbinden

- Das Einbinden von
  - Datei
  - Device
- Ist möglich als:
  - Scsi
  - Ide
  - Flash

Praxis: Ein erster Start

- Minimaler Aufruf:
  kvm Datei.image
- Vorgehensweise zum Installieren eines Windows-XP:
  - Erzeugen der Zielplatte z.B. mit
    ```
    # kvm-img create KvmPlatte.image 100G
    ```
  - Starten von CDRom:
    ```
    # kvm -m 512M -cdrom /dev/dvd -hda \
    KmvPlatte.image
    ```

Praxis: Mögliche Dateiformate

- KVM beherscht einfache, sparsame und kompatible Formate
  - Raw (ohne Formatierung) → **Sparse**-Dateien
  - Qcow2 (von qemu)
  - Vmdk 3&4 (vmware) → VMWare-Images laufen unverändert unter KVM
  - Cloop (Knoppix)

Praxis: Sparse-Dateien 1/2

- Bei Sparsedateien wird nur der tatsächlich belegte Speicherplatz verbraucht
- Wird von einigen Dateisystemen beherrscht:
  - Ext3 und Ext4
  - ReiserFS, Reiser4
  - NTFS
- Kann kein Sparse:
  - FAT, VFAT
  - XFS, JFS

Praxis: Sparse-Dateien 2/2

- Sparsedateien können nur auf geeigneten Dateisystemen erzeugt werden.
- Bestehende Images können in Sparse umgewandelt werden:
  cp --sparse=always Image.alt Image.neu
- Dabei werden nur mit Nullen gefüllte Sektoren als leer betrachtet und somit nicht geschrieben

Praxis: Die KVM/Qemu-Konsole

- Es gibt 3 Views im KVM-Terminal
  - Alt+Gr+1: Standardausgabe
  - Alt+Gr+2: KVM/Qemu-Konsole
  - Alt+Gr+3: Serielle Konsole
- Funktioniert mit allen Gastsystemen
- Unter Linux: Hohe Verwechslungsgefahr mit Alt+{F1,F2,F3} für die Linux-Terminals

Praxis: Linux unter KVM

- Beispiel: Aktuelle Knoppix-6.0
- /proc/cpuinfo: Qemu-CPU, kein vmx-Flag
- Lspci: Qemu-Devices
- Lsusb: USB-Devices sind standardmäßig (noch) deaktiviert
- „top“: Standarmäßig 128MB RAM

Praxis: Netzwerkkonzepte

```
# kvm mein.image
```

entspricht

```
# kvm -hda mein.image -net nic -net user
```

Praxis: Netzwerkkonzept -net user

- User mode network stack
- Vorteile
  - Benötigt keine Administrator-Previlegien
  - Kein eigenes sichtbares Device im Hostsystem nötig
  - Bei Bedarf interner kvm-dhcp-Server vorhanden
- Nachteile
  - Von außen nicht erreichbar
  - Kvm-Module müssen bereits laufen

Praxis: Netzwerkkonzept -net nic

- Network Interface Card
- Vorteile
  - Erzeugt Device im Gastsystem
  - Verbindung zu Vlan → direkt ansprechbar
- Nachteile
  - Rootprevilegien nötig
  - Direkt ansprechbar

Praxis: Netzwerkkonzept -net tap

- Tap Network Interface
- Vorteile
  - Eigenes Device (z.B. win0)
  - Device über Hostsystem dynamisch routbar
  - Start-/Stopscripten dynamisch konfigurierbar
- Nachteile
  - Tun-Modul notwendig (Problem vserver)
  - Noch etwas buggy

Praxis: Weitere Netzwerkkonzepte

- -net socket
  - Direktverbindungen zwischen KVMs
  - Schnelles UDP-Multicast
- -net vde
  - Virtueller Netzwerkswitch
  - Muss in KVM explizit einkompiliert werden
  - Noch sehr Development

Praxis: VirtIO-Devices 1/2

- VirtIO Netzwerkdevides existieren für Linux, Windows-{2k,XP,Vista}
- Sehr hohe Geschwindigkeit (>1GByte/s auf Core2 mit 2,0 GhZ)
- VirtIO Blockdevices im Moment nur für Linux
- 2.6-er Kernel mit Virtio-only ca. 1.4 Mbyte groß (64Bit)
- Boottime < 1s

Praxis: VirtIO Devices 2/2

- VirtIO-Balloon verteilt den vorhanden Speicher dynamisch auf die Linux-Gäste nach Bedarf
- VirtIO-Console ersetzt das getty-Device
- Mit „-pcidevice host=bus:dev.function“ reicht virtIO PCI-Geräte durch

Praxis: Non-persistence Mode

- Non-persistence-mode: -snapshot <datei>
  - Originalimage bleibt unangetastet
  - Änderungen nachträglich einpflegbar
- Änderungen einpflegen mit
  `kvm-img snapshot -a <datei>`

Praxis: nette Helfer für Linux

- Kvm bietet erweiterten Support für Linux
  - Kernel direkt bootbar mit -kernel
  - Kernelparameter mit -append
  - Initial Ramdisk mit -initrd
- Beispiel:

```
kvm -m 1G -smp 2 \
-kernel /boot/vmlinuz \
-append „root=/dev/vda \
init=/sbin/init“
```

Praxis: Sonstige Helfer

- Interner Samba-Server
- Localtime hart einstellbar ;-)
- Floppy-Image für Windwos-Installation

Ausblick: Was gar nicht geht

- 64bit Guests in 32Bit Hosts
- Non-x86 Guests in x86-Hosts (→ Qemui)
- Mehr als 2GB/Guest in 32Bit-Hosts

Ausblick: Was noch nicht geht

- Hot Migration
- Virtualisation in Guest (AMD teilweise)
- 3D-Beschleunigung in Windows Guests
- Direkte VGA Virtualisierung wie VMWare
- VirtIO in BSD-Guests

Fazit

- Überraschend stabiles System
- Annähernd 100% Leistung der virtualisierten Instanzen
- Beschränkung auf x86-Architektur nicht wirklich einschränkend
- Aufgrund von GPL wohl DER Durchstarter bei Virtualisierung

Fragen?