Virtualisierung mit Freier Software

Sascha Wilde

Intevation GmbH

13. November 2007

Teil I

Einleitung

Die Idee

- Moderne Computer langweilen sich die meiste Zeit
- Für viele Anwendungen ist mehr als ein System nötig
- Ein Computer der tut als wäre er viele

Die Idee

- Moderne Computer langweilen sich die meiste Zeit
- Für viele Anwendungen ist mehr als ein System nötig
- Ein Computer der tut als wäre er viele

Die Idee

- Moderne Computer langweilen sich die meiste Zeit
- Für viele Anwendungen ist mehr als ein System nötig
- Ein Computer der tut als wäre er viele

Was Virtualisierung *nicht* ist...

- **Simulation:**
  Exakte virtuelle Nachbildung von beliebiger Hardware.
  - Hardware-Entwicklung
  - Software-Entwicklung meist für noch nicht existente Hardware
  - Meist deutlich langsamer als echte Hardware
- **Emulation:**
  Nachbildung des Verhaltens von beliebiger Hardware aus Anwendungssicht.
  - Ersatz für nicht verfügbare (z.B. alte) Hardware
  - Bisweilen schneller als echte Hardware

Was Virtualisierung *nicht* ist...

- **Simulation:**
  Exakte virtuelle Nachbildung von beliebiger Hardware.
  - Hardware-Entwicklung
  - Software-Entwicklung meist für noch nicht existente Hardware
  - Meist deutlich langsamer als echte Hardware
- **Emulation:**
  Nachbildung des Verhaltens von beliebiger Hardware aus Anwendungssicht.
  - Ersatz für nicht verfügbare (z.B. alte) Hardware
  - Bisweilen schneller als echte Hardware

Was Virtualisierung *nicht* ist...

- **Simulation:**
  Exakte virtuelle Nachbildung von beliebiger Hardware.
  - Hardware-Entwicklung
  - Software-Entwicklung meist für noch nicht existente Hardware
  - Meist deutlich langsamer als echte Hardware
- **Emulation:**
  Nachbildung des Verhaltens von beliebiger Hardware aus Anwendungssicht.
  - Ersatz für nicht verfügbare (z.B. alte) Hardware
  - Bisweilen schneller als echte Hardware

Was Virtualisierung *nicht* ist...

- **Simulation:**
  Exakte virtuelle Nachbildung von beliebiger Hardware.
  - Hardware-Entwicklung
  - Software-Entwicklung meist für noch nicht existente Hardware
  - Meist deutlich langsamer als echte Hardware
- **Emulation:**
  Nachbildung des Verhaltens von beliebiger Hardware aus Anwendungssicht.
  - Ersatz für nicht verfügbare (z.B. alte) Hardware
  - Bisweilen schneller als echte Hardware

Was Virtualisierung ist...

- **Virtualisierung:**
  „Vervielfätigung“ vorhandener Hardware

Einsatzgebiete für Virtualisierung

Überall wo sich viel Hardware die meiste Zeit „langweilt“.

- Serversysteme
  - Trennung von Diensten
  - Besser als „change roots“
  - trotzdem Vorsicht: Studie von Tavis Ormandy
- Testsysteme
  - LiveCDs
  - komplexe Softwaresysteme
  - Betriebsysteme
  - „Staging“ Systeme
- Software Entwicklung
  - Betriebssystementwicklung
  - Kompatibilitätstests
  - definierte Umgebungen

Einsatzgebiete für Virtualisierung

Überall wo sich viel Hardware die meiste Zeit „langweilt“.

- Serversysteme
  - Trennung von Diensten
  - Besser als „change roots“
  - trotzdem Vorsicht: Studie von Tavis Ormandy
- Testsysteme
  - LiveCDs
  - komplexe Softwaresysteme
  - Betriebsysteme
  - „Staging“ Systeme
- Software Entwicklung
  - Betriebssystementwicklung
  - Kompatibilitätstests
  - definierte Umgebungen

Einsatzgebiete für Virtualisierung

Überall wo sich viel Hardware die meiste Zeit „langweilt“.

- Serversysteme
  - Trennung von Diensten
  - Besser als „change roots“
  - trotzdem Vorsicht: Studie von Tavis Ormandy
- Testsysteme
  - LiveCDs
  - komplexe Softwaresysteme
  - Betriebsysteme
  - „Staging“ Systeme
- Software Entwicklung
  - Betriebssystementwicklung
  - Kompatibilitätstests
  - definierte Umgebungen

Einsatzgebiete für Virtualisierung

Überall wo sich viel Hardware die meiste Zeit „langweilt“.

- Serversysteme
  - Trennung von Diensten
  - Besser als „change roots“
  - trotzdem Vorsicht: Studie von Tavis Ormandy
- Testsysteme
  - LiveCDs
  - komplexe Softwaresysteme
  - Betriebsysteme
  - „Staging“ Systeme
- Software Entwicklung
  - Betriebssystementwicklung
  - Kompatibilitätstests
  - definierte Umgebungen

Virtuelle Geräte: Paravirtualisierung oder Emulation

Physikalische Hardware ist oft nur einmal vorhanden, darum werden virtuelle Geräte benötigt.

Zwei Möglichkeiten:

- **Emulierte Geräte** verhalten sich wie „echte“: langsam
- **Paravirtualisierung** setzt Treiber im Gast voraus: performant

Virtuelle Geräte: Paravirtualisierung oder Emulation

Physikalische Hardware ist oft nur einmal vorhanden, darum werden virtuelle Geräte benötigt.
Zwei Möglichkeiten:

- **Emulierte Geräte** verhalten sich wie „echte“: langsam
- **Paravirtualisierung** setzt Treiber im Gast voraus: performant

Virtuelle Geräte: Paravirtualisierung oder Emulation

Physikalische Hardware ist oft nur einmal vorhanden, darum werden virtuelle Geräte benötigt.
Zwei Möglichkeiten:

- **Emulierte Geräte** verhalten sich wie „echte“: langsam
- **Paravirtualisierung** setzt Treiber im Gast voraus: performant

Verschiedene Ebenen der Virtualisierung

- Reine Emulation im Userland
- Kernel Unterstützung (Linux Kernelmodule, `CONFIG_PARAVIRT` in neueren Linux Kerneln) mit Hypervisor im Userland
- Spezieller Hypervisor Kernel
- Hardware gestützt (Intel VT-x und AMD Pacifica)

Die Grenzen sind fließend.

Verschiedene Ebenen der Virtualisierung

- Reine Emulation im Userland
- Kernel Unterstützung (Linux Kernelmodule, `CONFIG_PARAVIRT` in neueren Linux Kerneln) mit Hypervisor im Userland
- Spezieller Hypervisor Kernel
- Hardware gestützt (Intel VT-x und AMD Pacifica)

Die Grenzen sind fließend.

Verschiedene Ebenen der Virtualisierung

- Reine Emulation im Userland
- Kernel Unterstützung (Linux Kernelmodule, `CONFIG_PARAVIRT` in neueren Linux Kerneln) mit Hypervisor im Userland
- Spezieller Hypervisor Kernel
- Hardware gestützt (Intel VT-x und AMD Pacifica)

Die Grenzen sind fließend.

Verschiedene Ebenen der Virtualisierung

- Reine Emulation im Userland
- Kernel Unterstützung (Linux Kernelmodule, `CONFIG_PARAVIRT` in neueren Linux Kerneln) mit Hypervisor im Userland
- Spezieller Hypervisor Kernel
- Hardware gestützt (Intel VT-x und AMD Pacifica)

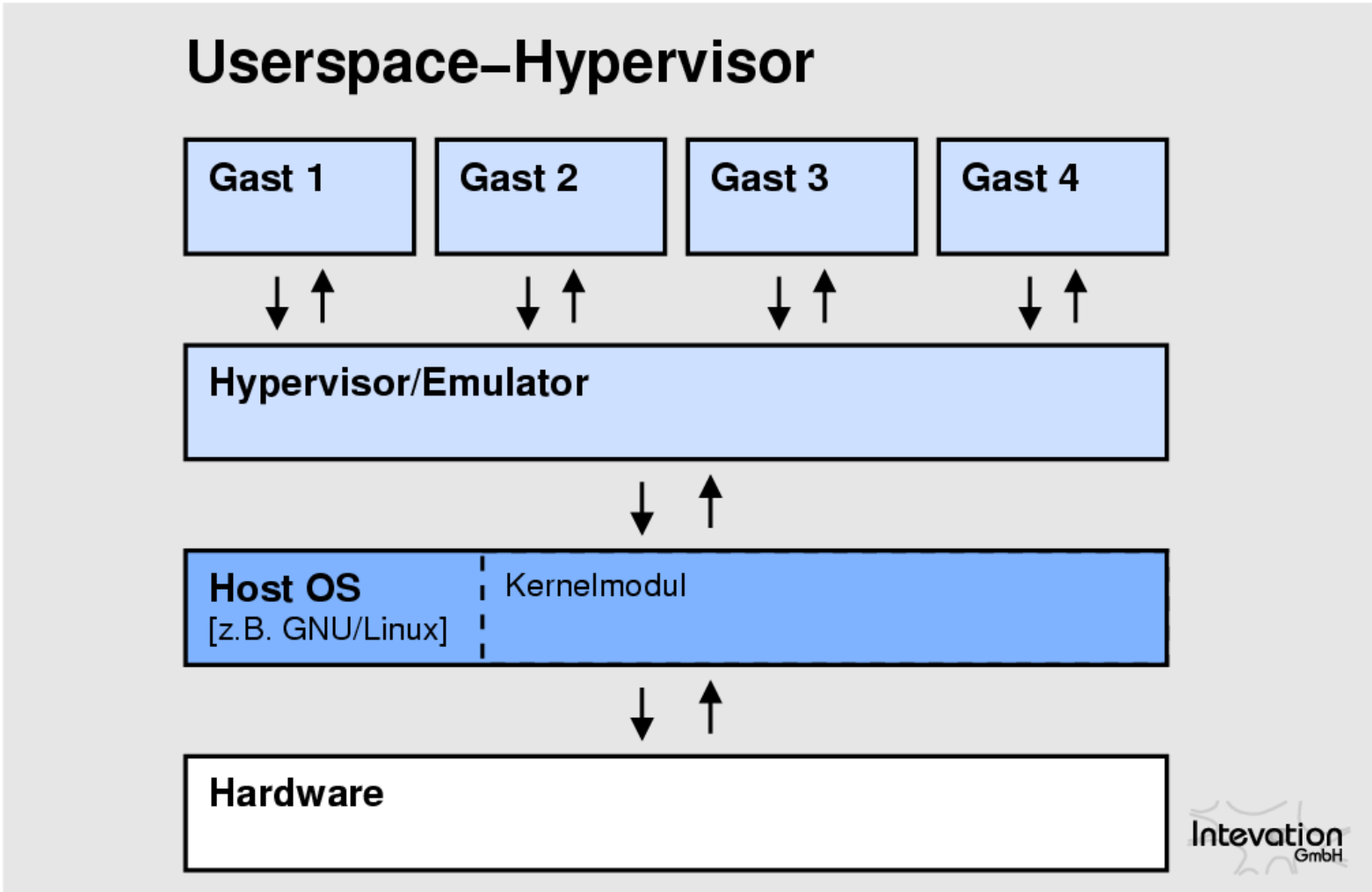
Die Grenzen sind fließend.

Verschiedene Ebenen der Virtualisierung

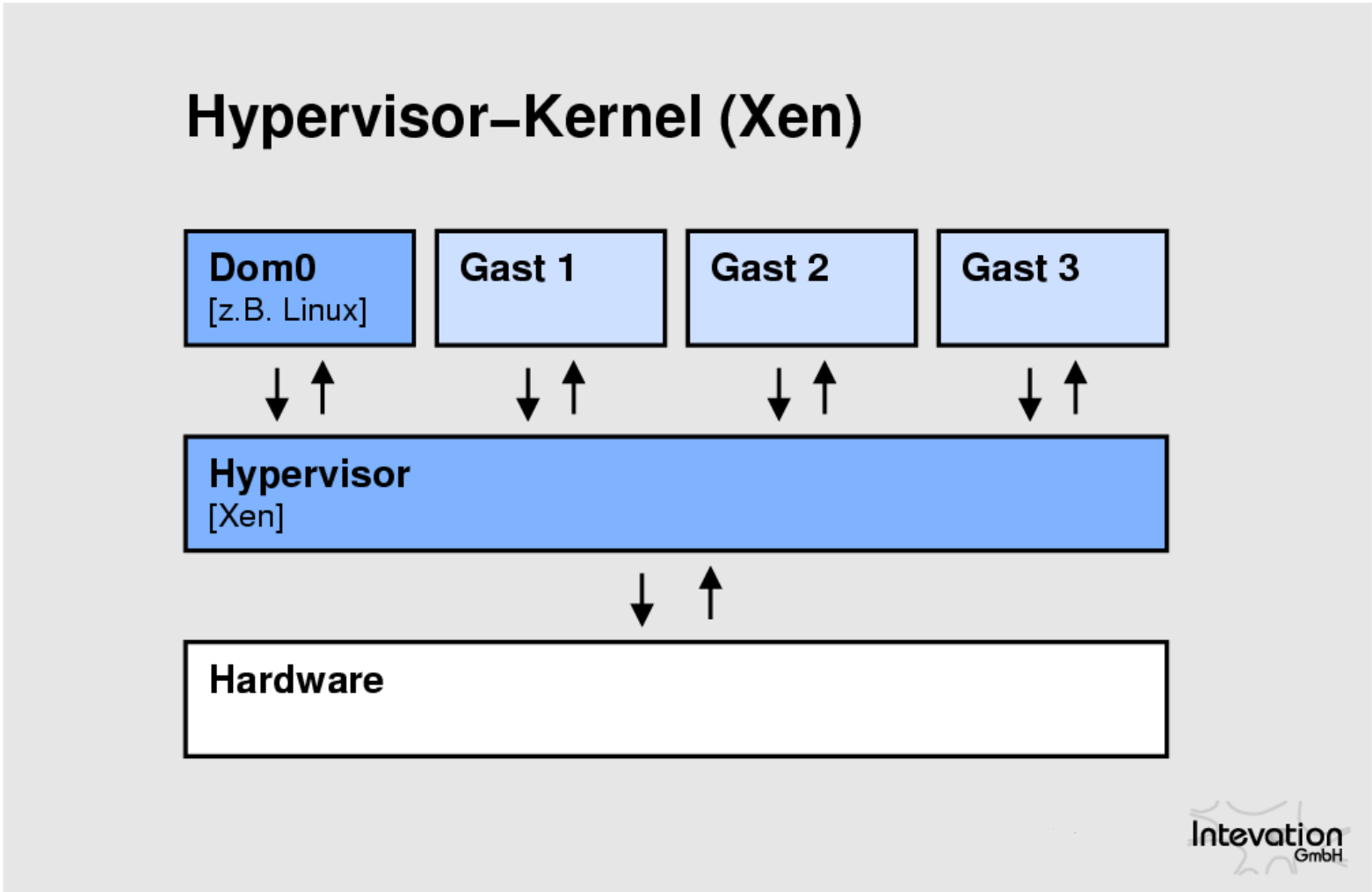
- Reine Emulation im Userland
- Kernel Unterstützung (Linux Kernelmodule, `CONFIG_PARAVIRT` in neueren Linux Kerneln) mit Hypervisor im Userland
- Spezieller Hypervisor Kernel
- Hardware gestützt (Intel VT-x und AMD Pacifica)

Die Grenzen sind fließend.

Hypervisor im Userspace



Hypervisor im Kernel



Verknüpfung physikalischer und realer Hardware im Wirt

- Platten/Partitionen
  - images/lvm/phys im Wirt
  - Partitionen oder Platten im Gast
- Netzwerk (nat, tun/tap, virtuelles Netz)

Verknüpfung physikalischer und realer Hardware im Wirt

- Platten/Partitionen
  - images/lvm/phys im Wirt
  - Partitionen oder Platten im Gast
- Netzwerk (nat, tun/tap, virtuelles Netz)

Teil II

Praxis

Untersuchte Virtualisierungssoftware

- Qemu
- VirtualBox
- Xen

Untersuchte Virtualisierungssoftware

- Qemu
- VirtualBox
- Xen

Untersuchte Virtualisierungssoftware

- Qemu
- VirtualBox
- Xen

Kurz gestreift

- KVM, Xen mit HVM
  beliebige Gäste da Hardware gestützte Virtualisierung mit emulierter Peripherie (Qemu)
- Lguest
  Sehr minimalistisch, verwendet neue Linux-Kernel-Feature, Hackerspielplatz

Kurz gestreift

- KVM, Xen mit HVM
  beliebige Gäste da Hardware gestützte Virtualisierung mit emulierter Peripherie (Qemu)
- Lguest
  Sehr minimalistisch, verwendet neue Linux-Kernel-Feature, Hackerspielplatz

Qemu mit Kqemu Kernelmodul

- Technologie: Emulation oder Emulation mit Kernelmodul
- Platten: Viele Imageformate,
  u.A. raw, qcow2, vmdk
- Netzwerk: NAT, TAP, pseudo-vlans, port redirect (userspace!)
- Interface: commandline, „monitor“ cli
- Hostsysteme: als reiner Emulation viele
  mit kqemu: GNU/Linux, Windows
- Gastsysteme: sehr flexibel
  (GNU/Linux, *BSD, Windows, Plan9, ...)

Qemu mit Kqemu Kernelmodul

- Technologie: Emulation oder Emulation mit Kernelmodul
- Platten: Viele Imageformate,
  u.A. raw, qcow2, vmdk
- Netzwerk: NAT, TAP, pseudo-vlans, port redirect (userspace!)
- Interface: commandline, „monitor“ cli
- Hostsysteme: als reiner Emulation viele
  mit kqemu: GNU/Linux, Windows
- Gastsysteme: sehr flexibel
  (GNU/Linux, *BSD, Windows, Plan9, ...)

Qemu mit Kqemu Kernelmodul

- Technologie: Emulation oder Emulation mit Kernelmodul
- Platten: Viele Imageformate,
  u.A. raw, qcow2, vmdk
- Netzwerk: NAT, TAP, pseudo-vlans, port redirect (userspace!)
- Interface: commandline, „monitor“ cli
- Hostsysteme: als reiner Emulation viele
  mit kqemu: GNU/Linux, Windows
- Gastsysteme: sehr flexibel
  (GNU/Linux, *BSD, Windows, Plan9, ...)

Qemu mit Kqemu Kernelmodul

- Technologie: Emulation oder Emulation mit Kernelmodul
- Platten: Viele Imageformate,
  u.A. raw, qcow2, vmdk
- Netzwerk: NAT, TAP, pseudo-vlans, port redirect (userspace!)
- Interface: commandline, „monitor“ cli
- Hostsysteme: als reiner Emulation viele
  mit kqemu: GNU/Linux, Windows
- Gastsysteme: sehr flexibel
  (GNU/Linux, *BSD, Windows, Plan9, ...)

Qemu mit Kqemu Kernelmodul

- Technologie: Emulation oder Emulation mit Kernelmodul
- Platten: Viele Imageformate,
  u.A. raw, qcow2, vmdk
- Netzwerk: NAT, TAP, pseudo-vlans, port redirect (userspace!)
- Interface: commandline, „monitor“ cli
- Hostsysteme: als reiner Emulation viele
  mit kqemu: GNU/Linux, Windows
- Gastsysteme: sehr flexibel
  (GNU/Linux, *BSD, Windows, Plan9, ...)

Qemu mit Kqemu Kernelmodul

- Technologie: Emulation oder Emulation mit Kernelmodul
- Platten: Viele Imageformate,
  u.A. raw, qcow2, vmdk
- Netzwerk: NAT, TAP, pseudo-vlans, port redirect (userspace!)
- Interface: commandline, „monitor“ cli
- Hostsysteme: als reiner Emulation viele
  mit kqemu: GNU/Linux, Windows
- Gastsysteme: sehr flexibel
  (GNU/Linux, *BSD, Windows, Plan9, ...)

Qemu mit Kqemu Kernelmodul

- **Pro**
  - User space (ggf. Kernelmodul)
  - Schlank und einfach
  - Wenig invasiv
  - Hohe Kompatibilität durch präzise Emulation
  - beliebige SMP Emulation
  - Headless möglich
  - VNC eingebaut
- **Kontra**
  - Relativ Langsam
  - Keine GUI
  - Netzwerk teilweise unflexibel

Qemu mit Kqemu Kernelmodul

- **Pro**
  - User space (ggf. Kernelmodul)
  - Schlank und einfach
  - Wenig invasiv
  - Hohe Kompatibilität durch präzise Emulation
  - beliebige SMP Emulation
  - Headless möglich
  - VNC eingebaut
- **Kontra**
  - Relativ Langsam
  - Keine GUI
  - Netzwerk teilweise unflexibel

Anwendungsgebiete:

- Entwicklung (low level)
- Software Tests
- Demos (keine/minimale Installation)

VirtualBox

- Technologie: Emulation mit JIT Code Modification und Kernelmodul
  Userspace daemon
  optional: „guest-additions“ (Grafik, time sync)
  benutzt qemu Devices
  optional: HW Support
- Platten: vdi, vmdk (converter für raw)
- Netzwerk: NAT, tap, pseudo vlan
- Interface: GUI (Qt), und Kommandozeile (kein headless!)
- Hostsysteme: GNU/Linux, Windows, MacOS X
- Gastsysteme: GNU/Linux, Windows, Solaris, Andere?

VirtualBox

- Technologie: Emulation mit JIT Code Modification und Kernelmodul
  Userspace daemon
  optional: „guest-additions“ (Grafik, time sync)
  benutzt qemu Devices
  optional: HW Support
- Platten: vdi, vmdk (converter für raw)
- Netzwerk: NAT, tap, pseudo vlan
- Interface: GUI (Qt), und Kommandozeile (kein headless!)
- Hostsysteme: GNU/Linux, Windows, MacOS X
- Gastsysteme: GNU/Linux, Windows, Solaris, Andere?

VirtualBox

- Technologie: Emulation mit JIT Code Modification und Kernelmodul
  Userspace daemon
  optional: „guest-additions“ (Grafik, time sync)
  benutzt qemu Devices
  optional: HW Support
- Platten: vdi, vmdk (converter für raw)
- Netzwerk: NAT, tap, pseudo vlan
- Interface: GUI (Qt), und Kommandozeile (kein headless!)
- Hostsysteme: GNU/Linux, Windows, MacOS X
- Gastsysteme: GNU/Linux, Windows, Solaris, Andere?

VirtualBox

- Technologie: Emulation mit JIT Code Modification und Kernelmodul
  Userspace daemon
  optional: „guest-additions“ (Grafik, time sync)
  benutzt qemu Devices
  optional: HW Support
- Platten: vdi, vmdk (converter für raw)
- Netzwerk: NAT, tap, pseudo vlan
- Interface: GUI (Qt), und Kommandozeile (kein headless!)
- Hostsysteme: GNU/Linux, Windows, MacOS X
- Gastsysteme: GNU/Linux, Windows, Solaris, Andere?

VirtualBox

- Technologie: Emulation mit JIT Code Modification und Kernelmodul
  Userspace daemon
  optional: „guest-additions“ (Grafik, time sync)
  benutzt qemu Devices
  optional: HW Support
- Platten: vdi, vmdk (converter für raw)
- Netzwerk: NAT, tap, pseudo vlan
- Interface: GUI (Qt), und Kommandozeile (kein headless!)
- Hostsysteme: GNU/Linux, Windows, MacOS X
- Gastsysteme: GNU/Linux, Windows, Solaris, Andere?

VirtualBox

- Technologie: Emulation mit JIT Code Modification und Kernelmodul
  Userspace daemon
  optional: „guest-additions“ (Grafik, time sync)
  benutzt qemu Devices
  optional: HW Support
- Platten: vdi, vmdk (converter für raw)
- Netzwerk: NAT, tap, pseudo vlan
- Interface: GUI (Qt), und Kommandozeile (kein headless!)
- Hostsysteme: GNU/Linux, Windows, MacOS X
- Gastsysteme: GNU/Linux, Windows, Solaris, Andere?

VirtualBox

- **Pro**
  - User space (+ nur ein Kernelmodul)
  - GUI
  - Kann VMware Images direkt lesen (aber nicht die VM Konfig)
- **Kontra**
  - Manchmal Langsam
  - Immer GUI Ausgabe des Gastes (in der freien Version)
  - Noch keine Binärpakete der freien Version

VirtualBox

- **Pro**
  - User space (+ nur ein Kernelmodul)
  - GUI
  - Kann VMware Images direkt lesen (aber nicht die VM Konfig)
- **Kontra**
  - Manchmal Langsam
  - Immer GUI Ausgabe des Gastes (in der freien Version)
  - Noch keine Binärpakete der freien Version

VirtualBox Anwendungsgebiete:

- Software Tests
- Demos

Xen3

- Technologie: Dedizierter Hypervisor Kernel Paravirtualisierung mit Device Treibern des DOM0 Kernels (Treiber im Gast!)
- optional: HW Support, dann volle Virtualisierung (mit qemu Devices)
- Platten: Geräte, Partitionen oder Images (je Partition!)
- Netzwerk: TAP, pseudo vlan, (sehr Flexibel per Scripte: nat, bridging, routing)
- Interface: Kommandozeile, Konfigurations Files (flexibel, da Python)
- GUIs verfügbar
- Hostsysteme: GNU/Linux, NetBSD
- Gastsysteme: GNU/Linux, NetBSD (experimentell andere) Mit HVM: Windows, Andere?

Xen3

- Technologie: Dedizierter Hypervisor Kernel Paravirtualisierung mit Device Treibern des DOM0 Kernels (Treiber im Gast!)
- optional: HW Support, dann volle Virtualisierung (mit qemu Devices)
- Platten: Geräte, Partitionen oder Images (je Partition!)
- Netzwerk: TAP, pseudo vlan, (sehr Flexibel per Scripte: nat, bridging, routing)
- Interface: Kommandozeile, Konfigurations Files (flexibel, da Python)
- GUIs verfügbar
- Hostsysteme: GNU/Linux, NetBSD
- Gastsysteme: GNU/Linux, NetBSD (experimentell andere) Mit HVM: Windows, Andere?

Xen3

- Technologie: Dedizierter Hypervisor Kernel Paravirtualisierung mit Device Treibern des DOM0 Kernels (Treiber im Gast!)
- optional: HW Support, dann volle Virtualisierung (mit qemu Devices)
- Platten: Geräte, Partitionen oder Images (je Partition!)
- Netzwerk: TAP, pseudo vlan, (sehr Flexibel per Scripte: nat, bridging, routing)
- Interface: Kommandozeile, Konfigurations Files (flexibel, da Python)
- GUIs verfügbar
- Hostsysteme: GNU/Linux, NetBSD
- Gastsysteme: GNU/Linux, NetBSD (experimentell andere) Mit HVM: Windows, Andere?

Xen3

- Technologie: Dedizierter Hypervisor Kernel Paravirtualisierung mit Device Treibern des DOM0 Kernels (Treiber im Gast!)
- optional: HW Support, dann volle Virtualisierung (mit qemu Devices)
- Platten: Geräte, Partitionen oder Images (je Partition!)
- Netzwerk: TAP, pseudo vlan, (sehr Flexibel per Scripte: nat, bridging, routing)
- Interface: Kommandozeile, Konfigurations Files (flexibel, da Python)
- GUIs verfügbar
- Hostsysteme: GNU/Linux, NetBSD
- Gastsysteme: GNU/Linux, NetBSD (experimentell andere) Mit HVM: Windows, Andere?

Xen3

- Technologie: Dedizierter Hypervisor Kernel
  Paravirtualisierung mit Device Treibern des DOM0 Kernels (Treiber im Gast!)
- optional: HW Support, dann volle Virtualisierung (mit qemu Devices)
- Platten: Geräte, Partitionen oder Images (je Partition!)
- Netzwerk: TAP, pseudo vlan, (sehr Flexibel per Scripte: nat, bridging, routing)
- Interface: Kommandozeile, Konfigurations Files (flexibel, da Python)
- GUIs verfügbar
- Hostsysteme: GNU/Linux, NetBSD
- Gastsysteme: GNU/Linux, NetBSD (experimentell andere)
  Mit HVM: Windows, Andere?

Xen3

- Technologie: Dedizierter Hypervisor Kernel Paravirtualisierung mit Device Treibern des DOM0 Kernels (Treiber im Gast!)
- optional: HW Support, dann volle Virtualisierung (mit qemu Devices)
- Platten: Geräte, Partitionen oder Images (je Partition!)
- Netzwerk: TAP, pseudo vlan, (sehr Flexibel per Scripte: nat, bridging, routing)
- Interface: Kommandozeile, Konfigurations Files (flexibel, da Python)
- GUIs verfügbar
- Hostsysteme: GNU/Linux, NetBSD
- Gastsysteme: GNU/Linux, NetBSD (experimentell andere) Mit HVM: Windows, Andere?

Xen3

- Technologie: Dedizierter Hypervisor Kernel Paravirtualisierung mit Device Treibern des DOM0 Kernels (Treiber im Gast!)
- optional: HW Support, dann volle Virtualisierung (mit qemu Devices)
- Platten: Geräte, Partitionen oder Images (je Partition!)
- Netzwerk: TAP, pseudo vlan, (sehr Flexibel per Scripte: nat, bridging, routing)
- Interface: Kommandozeile, Konfigurations Files (flexibel, da Python)
- GUIs verfügbar
- Hostsysteme: GNU/Linux, NetBSD
- Gastsysteme: GNU/Linux, NetBSD (experimentell andere) Mit HVM: Windows, Andere?

Xen3

- Technologie: Dedizierter Hypervisor Kernel
  Paravirtualisierung mit Device Treibern des DOM0 Kernels (Treiber im Gast!)
- optional: HW Support, dann volle Virtualisierung (mit qemu Devices)
- Platten: Geräte, Partitionen oder Images (je Partition!)
- Netzwerk: TAP, pseudo vlan, (sehr Flexibel per Scripte: nat, bridging, routing)
- Interface: Kommandozeile, Konfigurations Files (flexibel, da Python)
- GUIs verfügbar
- Hostsysteme: GNU/Linux, NetBSD
- Gastsysteme: GNU/Linux, NetBSD (experimentell andere)
  Mit HVM: Windows, Andere?

Xen3 – Pro und Kontra

- **Pro**
  - Sehr Performant
  - Relativ Sicher
  - Sehr flexible Konfiguration
- **Kontra**
  - Aufwndiges Setup (Hypervisor-Kernel + Dediziertes DOM0 System)
  - Rootrechte unabdingbar
  - Keine grafische Ausgabe

Xen3 – Pro und Kontra

- **Pro**
  - Sehr Performant
  - Relativ Sicher
  - Sehr flexible Konfiguration
- **Kontra**
  - Aufwndiges Setup (Hypervisor-Kernel + Dediziertes DOM0 System)
  - Rootrechte unabdingbar
  - Keine grafische Ausgabe

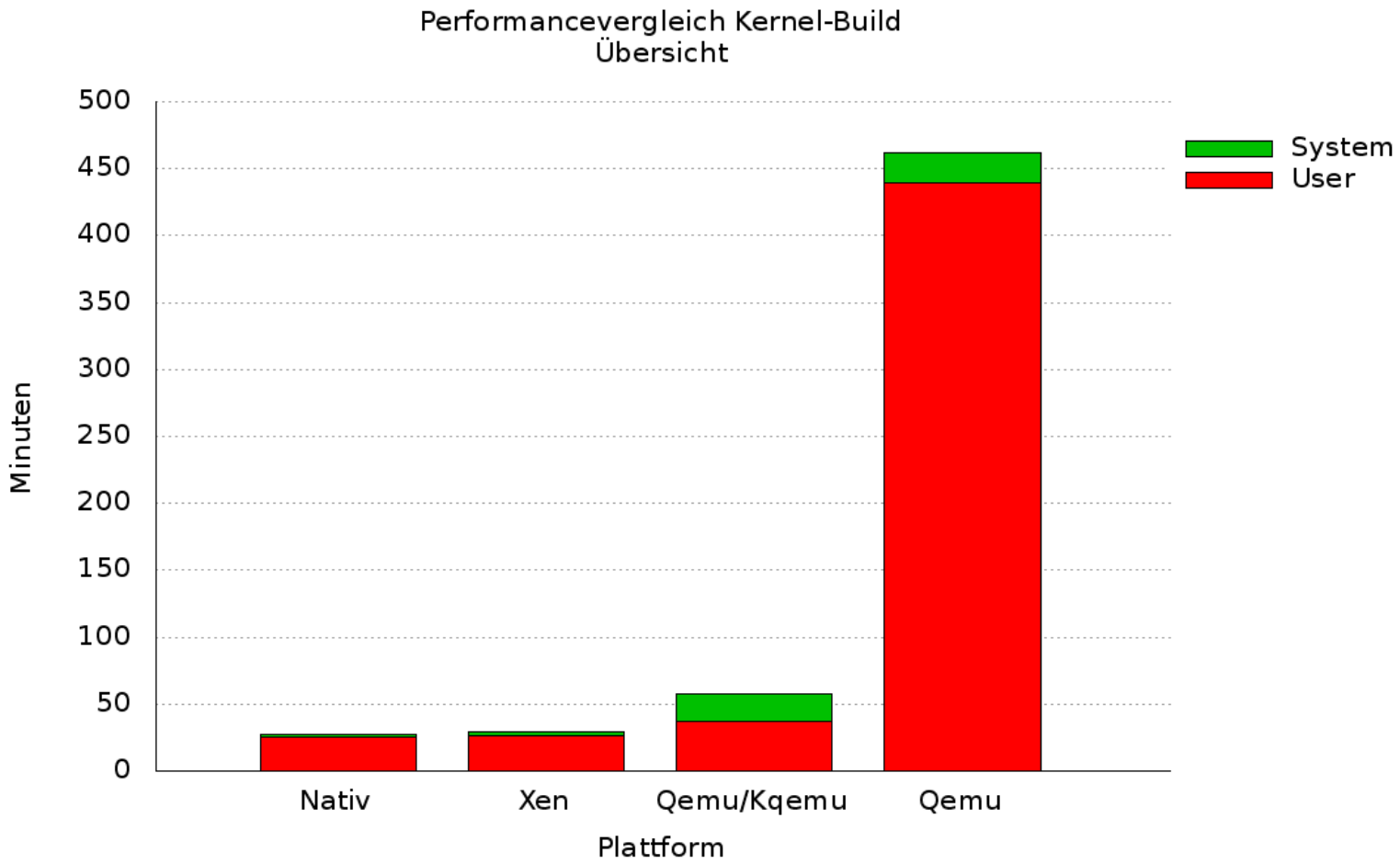
Anwendungsgebiete:

- Server Virtualisierung
- Highlevel Development (Bau-Systeme)
- Staging
- Software Tests

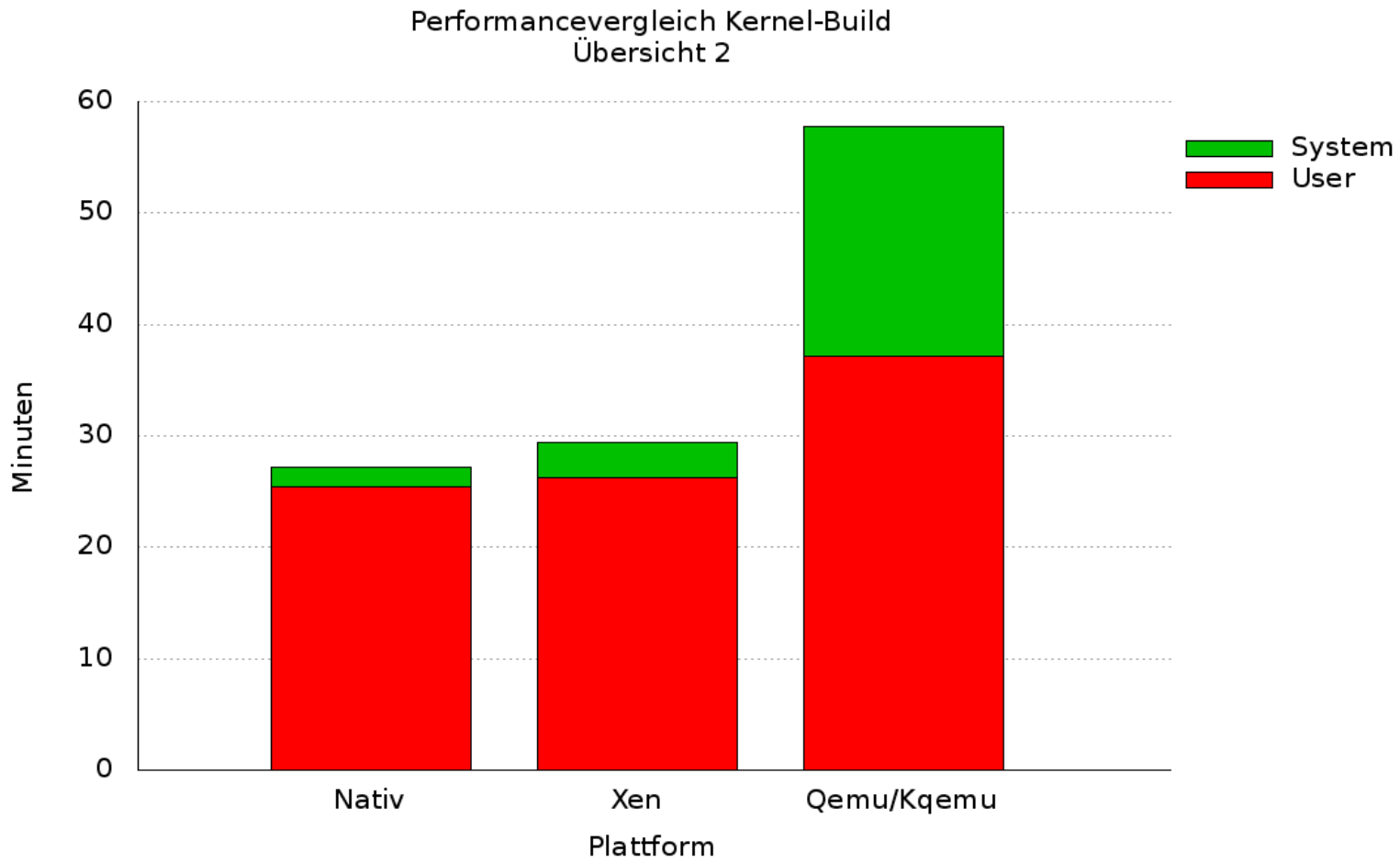
Teil III

Performace

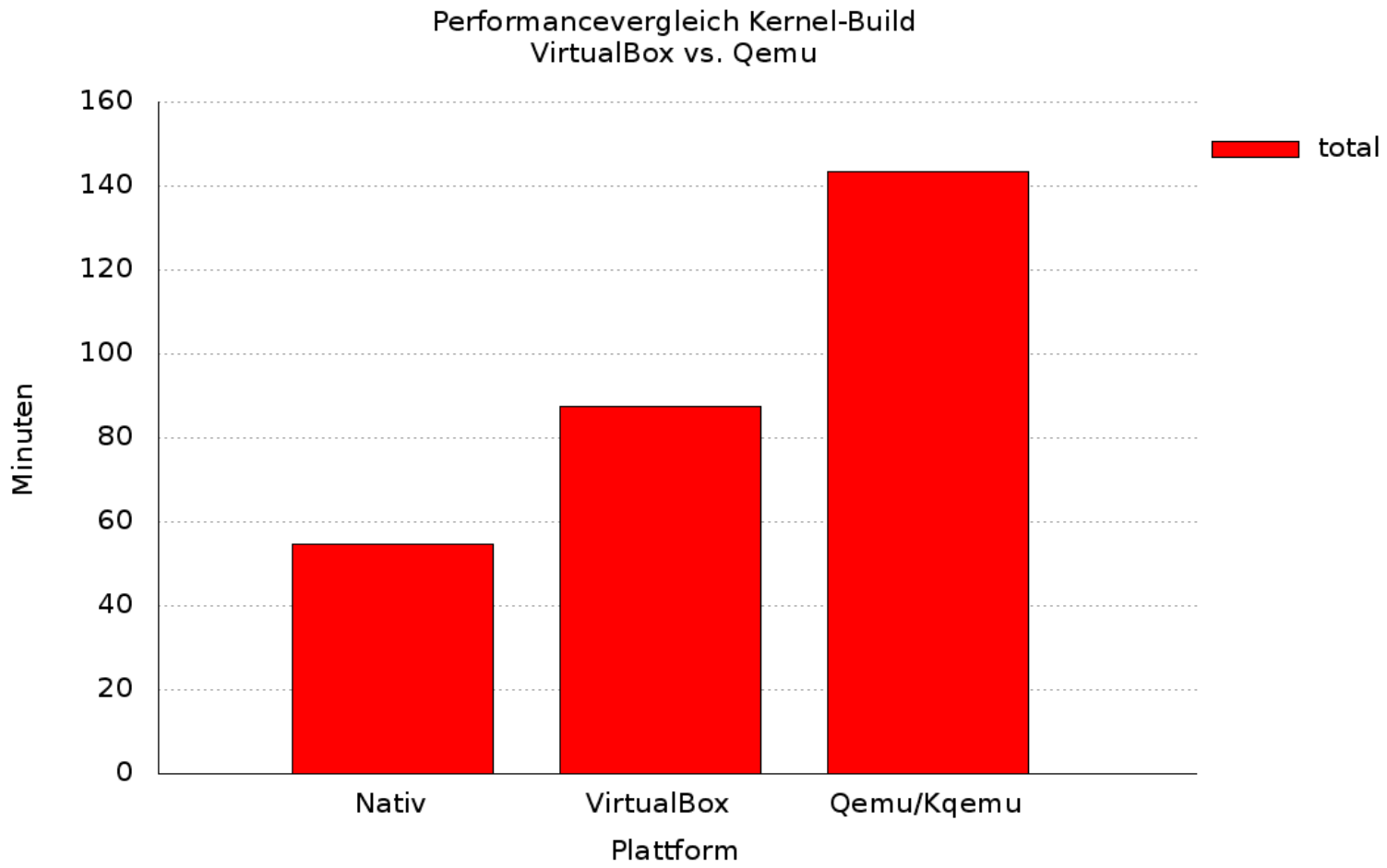
Xen3 vs. Qemu mit und ohne Kernelmodul



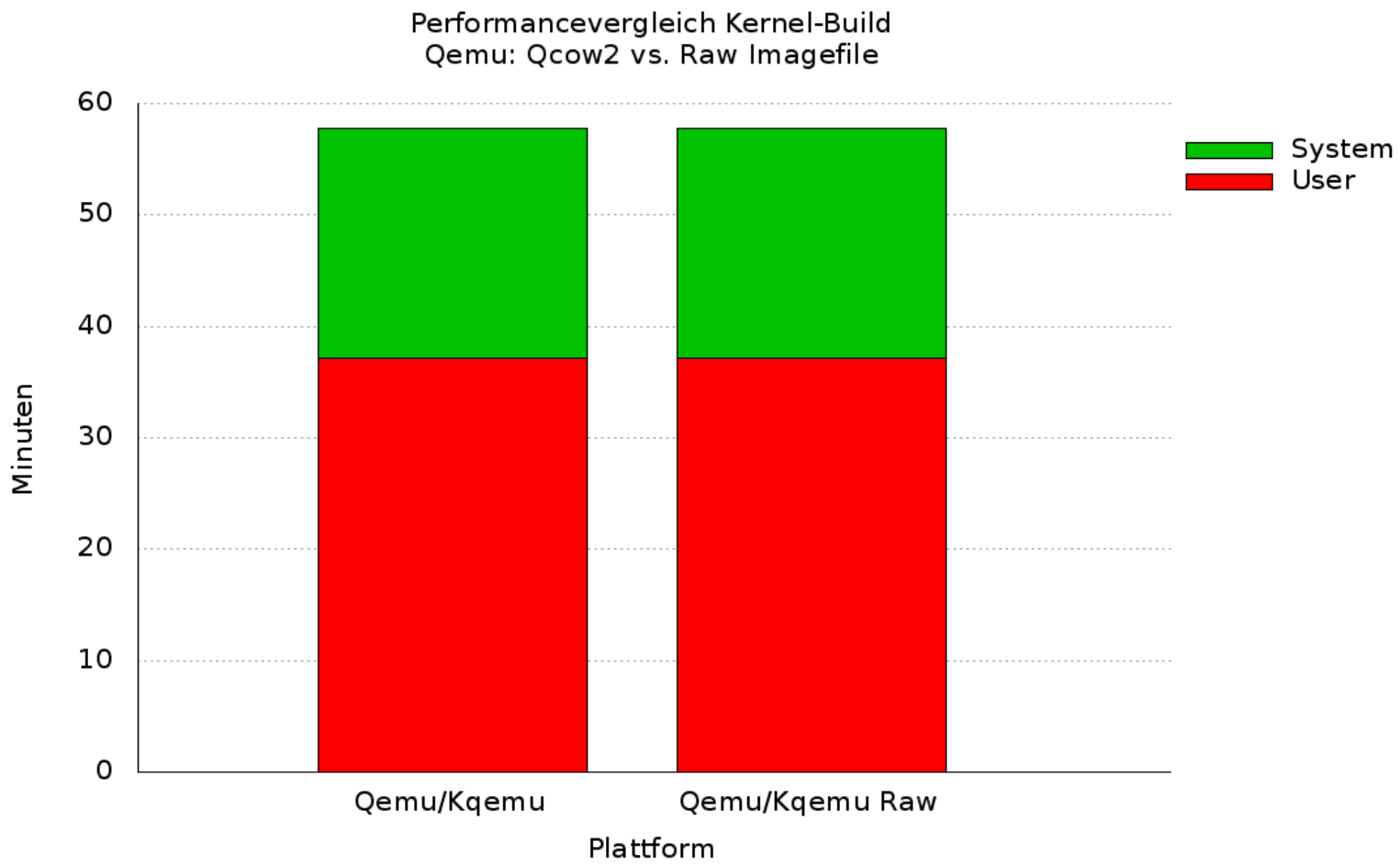
Xen3 vs. Qemu mit Kernelmodul



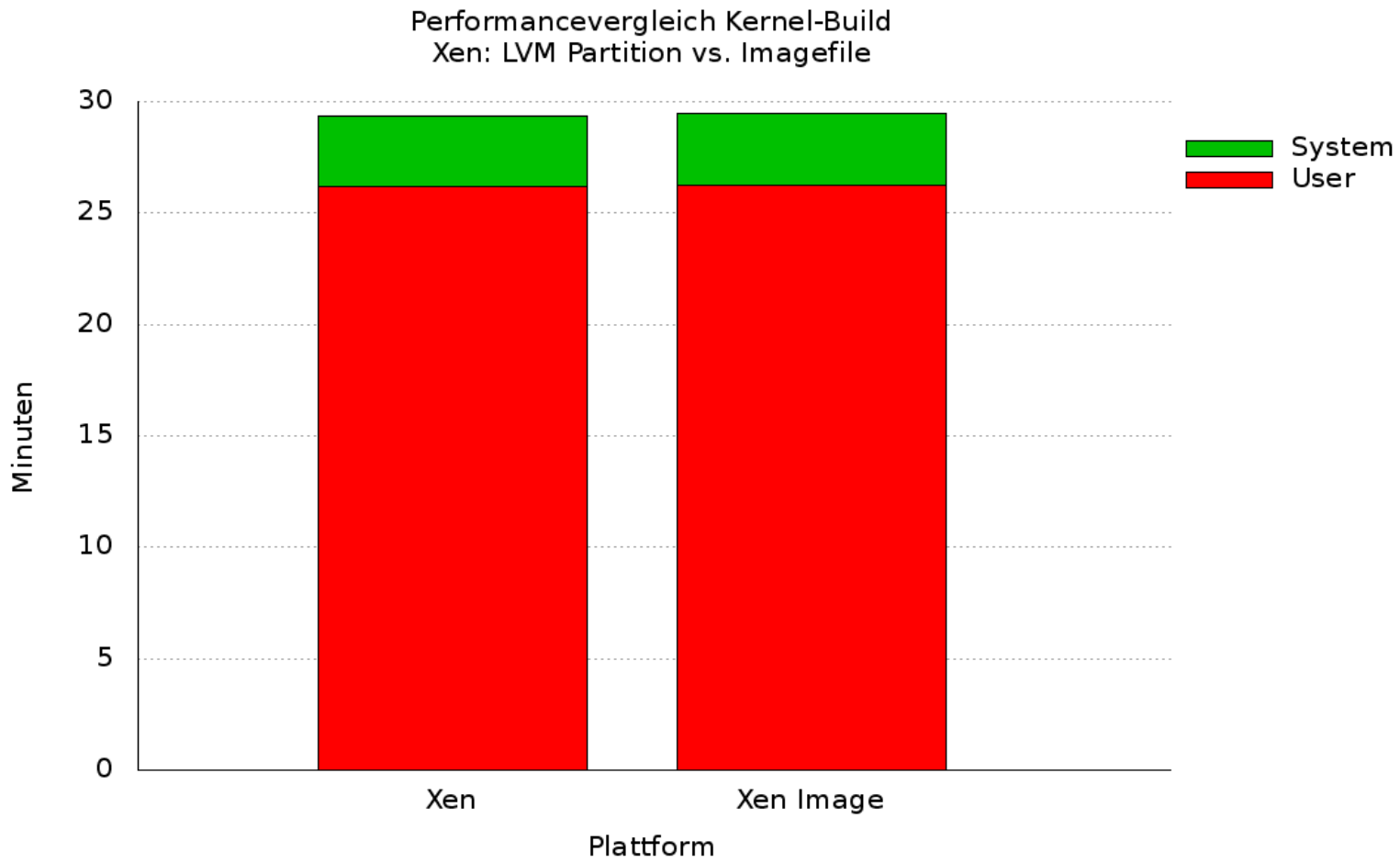
VirtualBox vs. Qemu mit Kernelmodul



Vergleich Qemu Imageformate



Vergleich Xen3 native Partition mit Imagefile



Teil IV

Demonstration

Teil V

Quellen

Weiterführende Links und Quellen

- http://fabrice.bellard.free.fr/qemu/
- http://www.virtualbox.org/
- http://www.xen.org/
- http://taviso.decsystem.org/virtsec.pdf
- http://googleonlinesecurity.blogspot.com/2007/05/on-virtualisation.html