Xen
- die Kunst der Virtualisierung

Dr. Carsten Gnörlich
cg@techfak.uni-bielefeld.de

26.06.2007

Aufbau des Vortrags

1. Ein bißchen Theorie (ca. 30min)
2. Xen-Live-Demo (ca. ?? - Murphy ;-)
3. Zusammenfassung (ca. 10 min)

Das große Bild, Teil 1

Hardware-Virtualisierung:

- “Wirt” und “Gast” sind jeweils komplette Betriebssysteme

➞ mehrere Gäste (Betriebsysteme) auf einer Hardware

➞ Gäste gegeneinander abschotten

Applikations-Virtualisierung (hier *nicht* behandelt):

- auf Programm/Thread–Ebene; Beispiel: Java-VM

Stufen der Virtualisierung (1)

keine

DOS

Stufen der Virtualisierung (2)

keine	CPU-Zeit	- Scheduler
	Speicher	- virtueller Speicher

DOS Unix

Stufen der Virtualisierung (3)

keine	CPU-Zeit	Dateiraum
	Speicher	Prozeßraum
DOS	Unix	chroot
		vserver
		zones

Trennung durch Kernel-Features

Wirt und alle Gäste laufen unter dem gleichen Kernel

Stufen der Virtualisierung (4)

keine	CPU-Zeit	Dateiraum	Hardware-Virtualisierung
	Speicher	Prozeßraum	
			Paravirtualisierung (Software)
DOS	Unix	chroot	Xen-Kernel
		vserver	
		zones	- gleiche Arch., aber unterschiedl. Kernel für Wirt/Gast
			- modifizierter Gast-Kernel

Stufen der Virtualisierung (5)

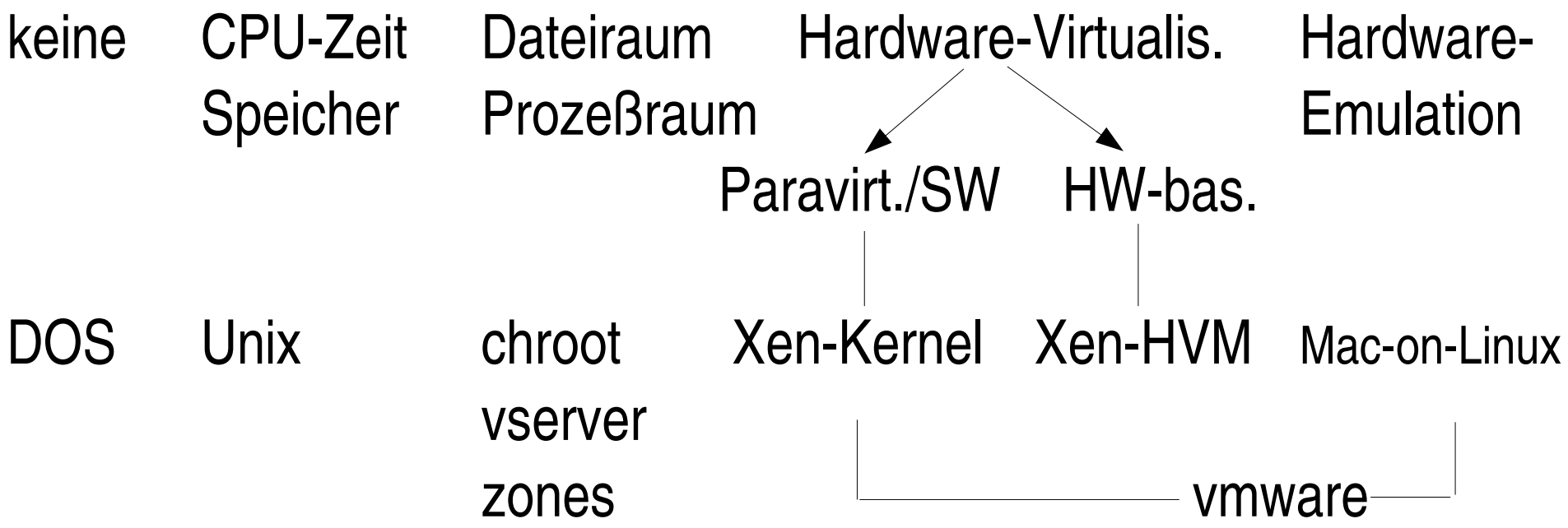
keine	CPU-Zeit	Dateiraum	Hardware-Virtualisierung
	Speicher	Prozeßraum	
			Paravirt./SW HW-basiert
DOS	Unix	chroot	Xen-Kernel Xen-HVM
		vserver	
		zones	- unmodifizierter Gastkernel
			- braucht HW-Unterstützung in CPU

Stufen der Virtualisierung (6)

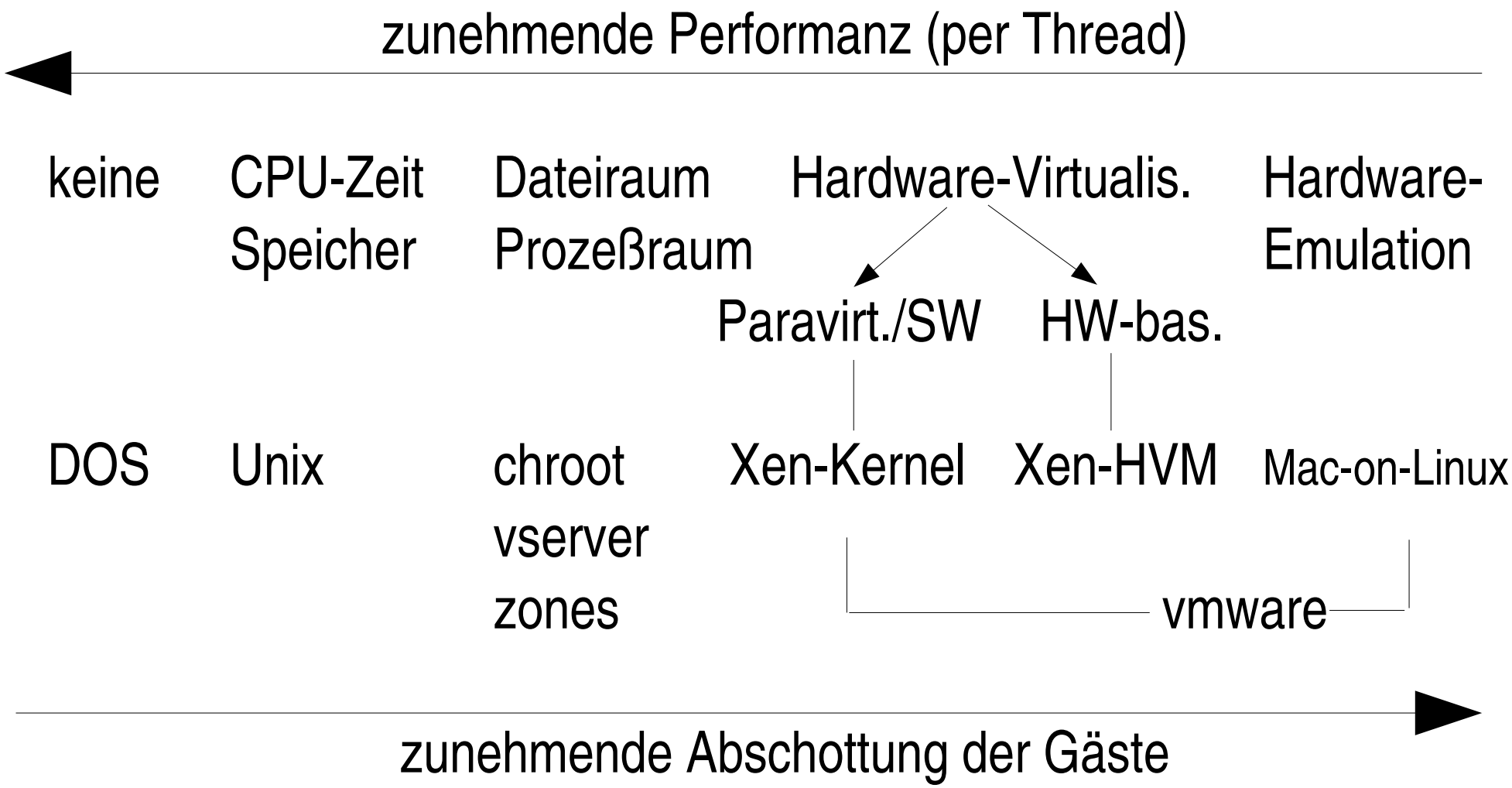
keine	CPU-Zeit Speicher	Dateiraum Prozeßraum	Hardware-Virtualis.		Hardware- Emulation
			Paravirt./SW	HW-bas.	
DOS	Unix	chroot vserver zones	Xen-Kernel	Xen-HVM	Mac-on-Linux

- unterschiedliche Architektur für Wirt und Gast
- Bsp: PPC-Mac auf x86-Linux

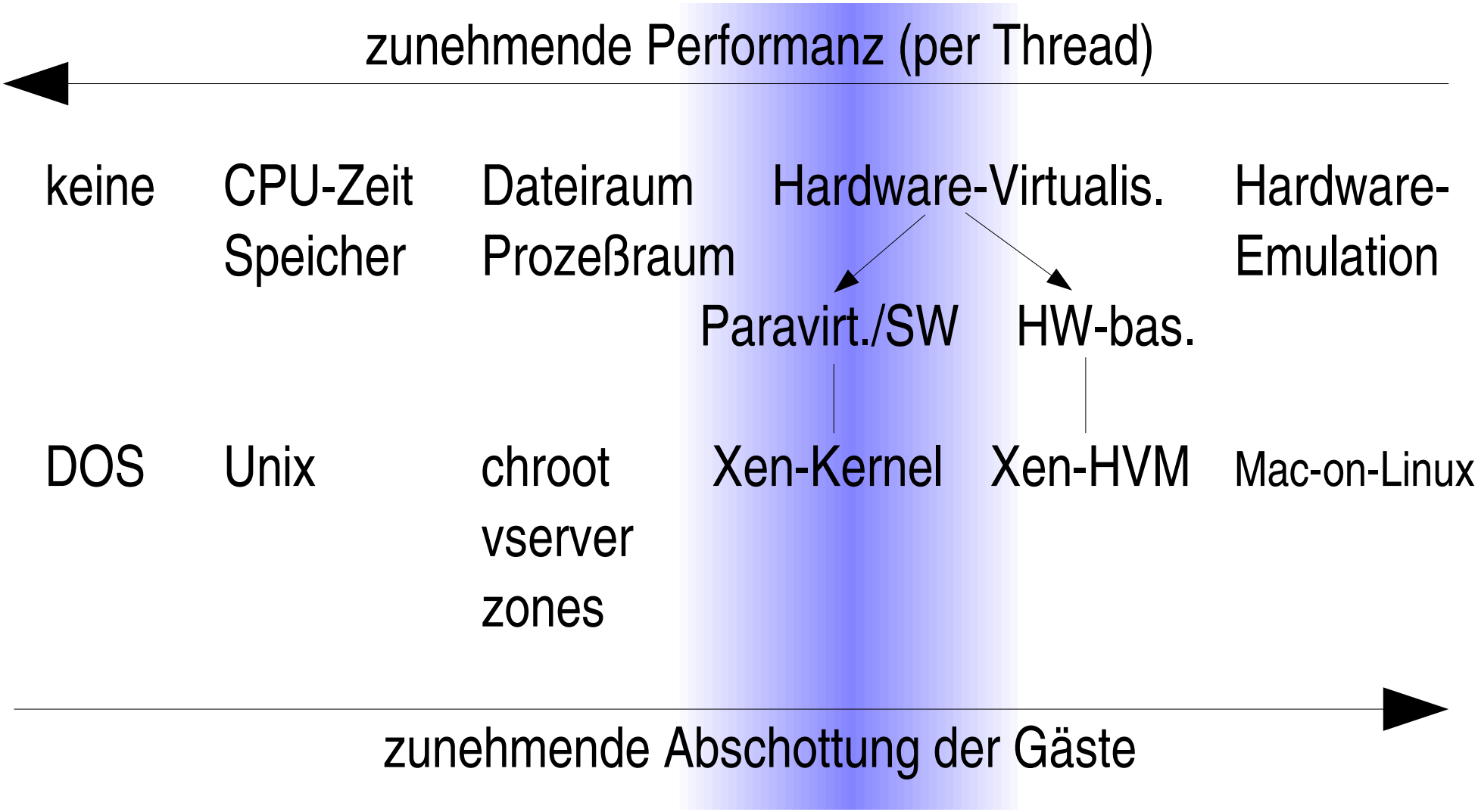
Stufen der Virtualisierung (7)



Stufen der Virtualisierung (8)

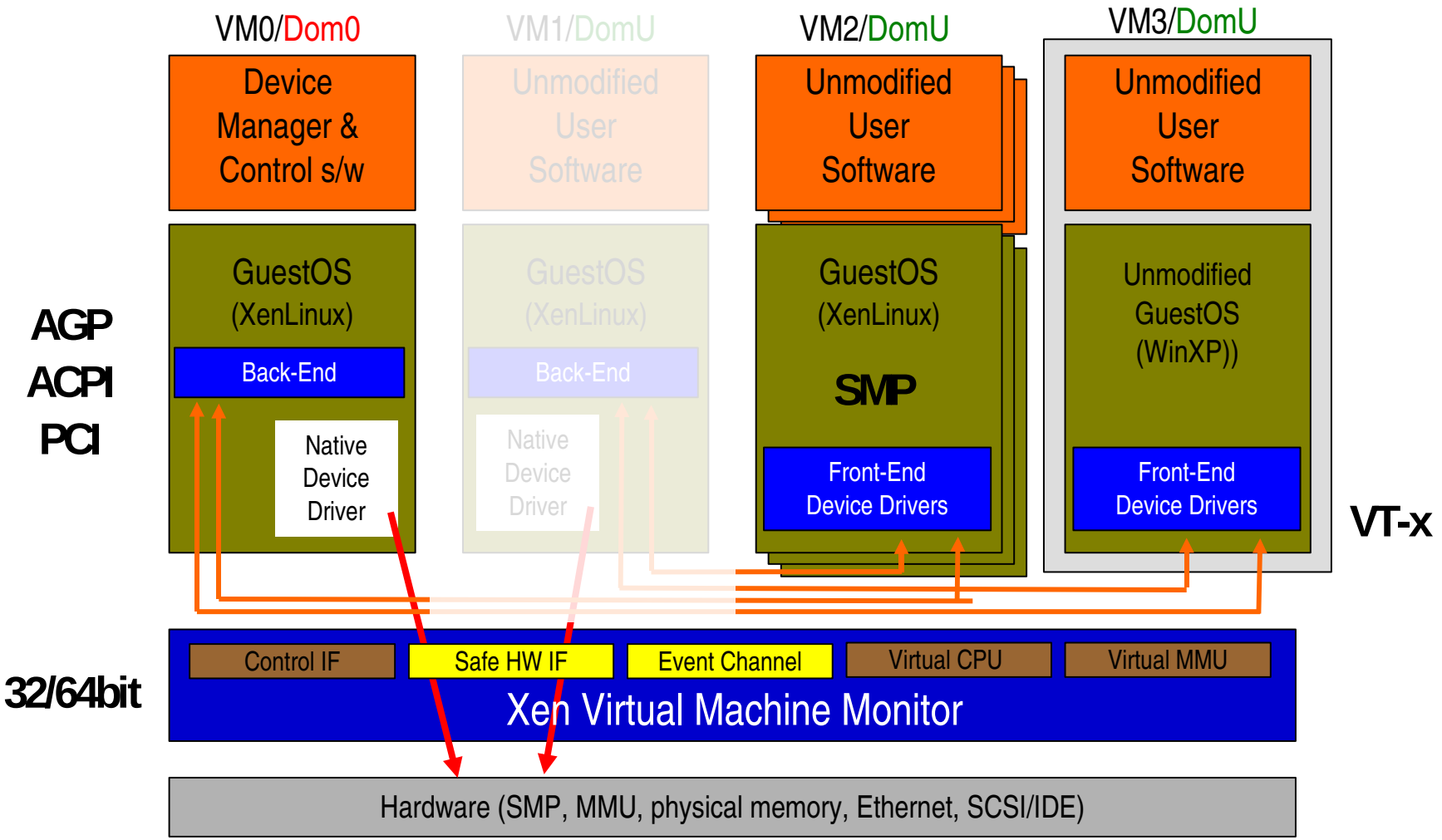


“Sweet spot”

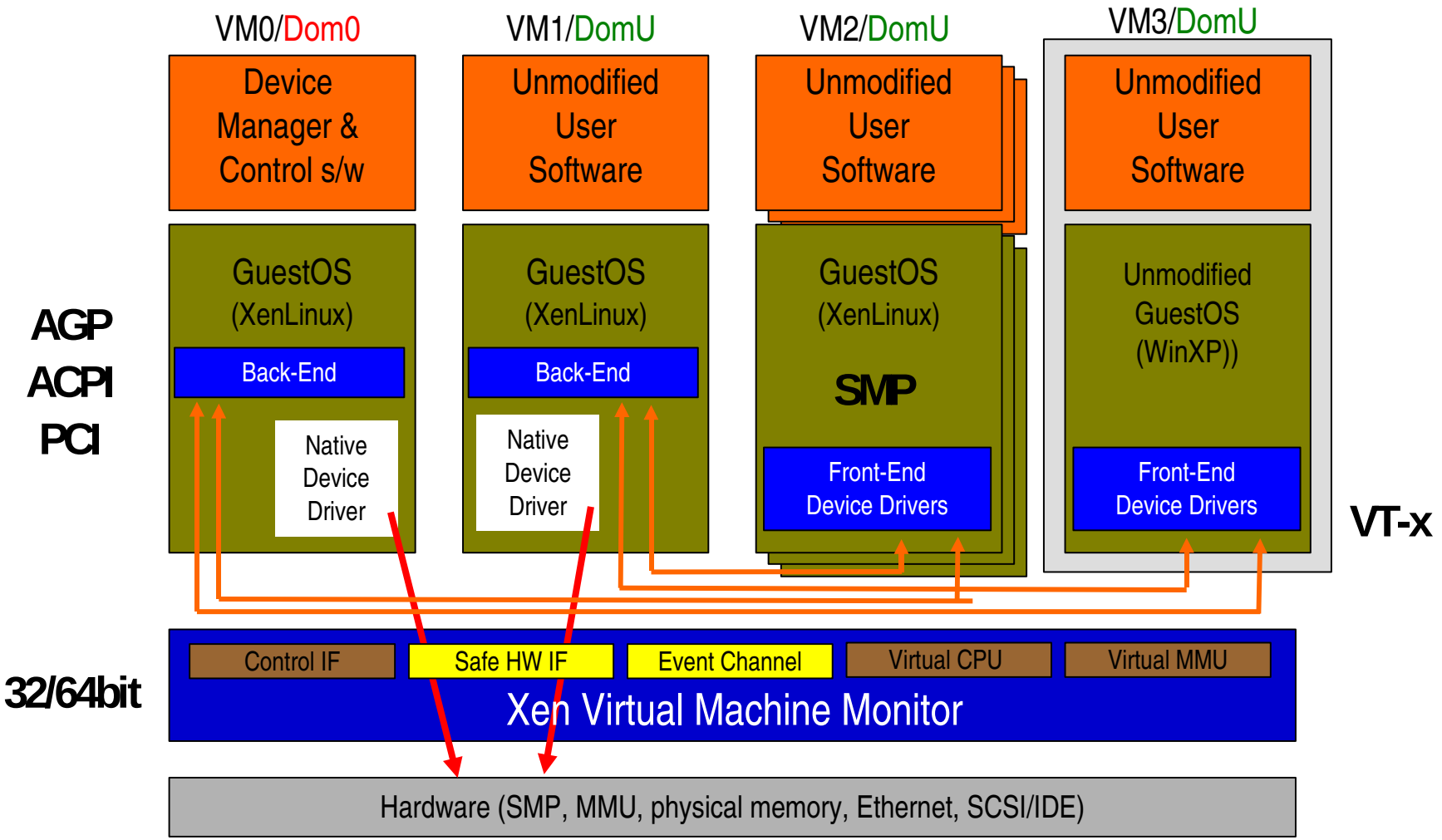

zunehmende Performanz (per Thread)
keine
CPU-Zeit
Speicher
Dateiraum
Prozeßraum
Hardware-Virtualis.
Hardware-
Emulation
Paravirt./SW
HW-bas.
DOS
Unix
chroot
vserver
zones
Xen-Kernel
Xen-HVM
Mac-on-Linux
zunehmende Abschottung der Gäste

Virtualisierung und Paravirtualisierung

Das große Bild, Teil 2



Das große Bild, Teil 2



Isolation der virtuellen Maschinen untereinander

- klassischer “Unix”-Zugriffsschutz

  Prozeß- / Filespaces, ...

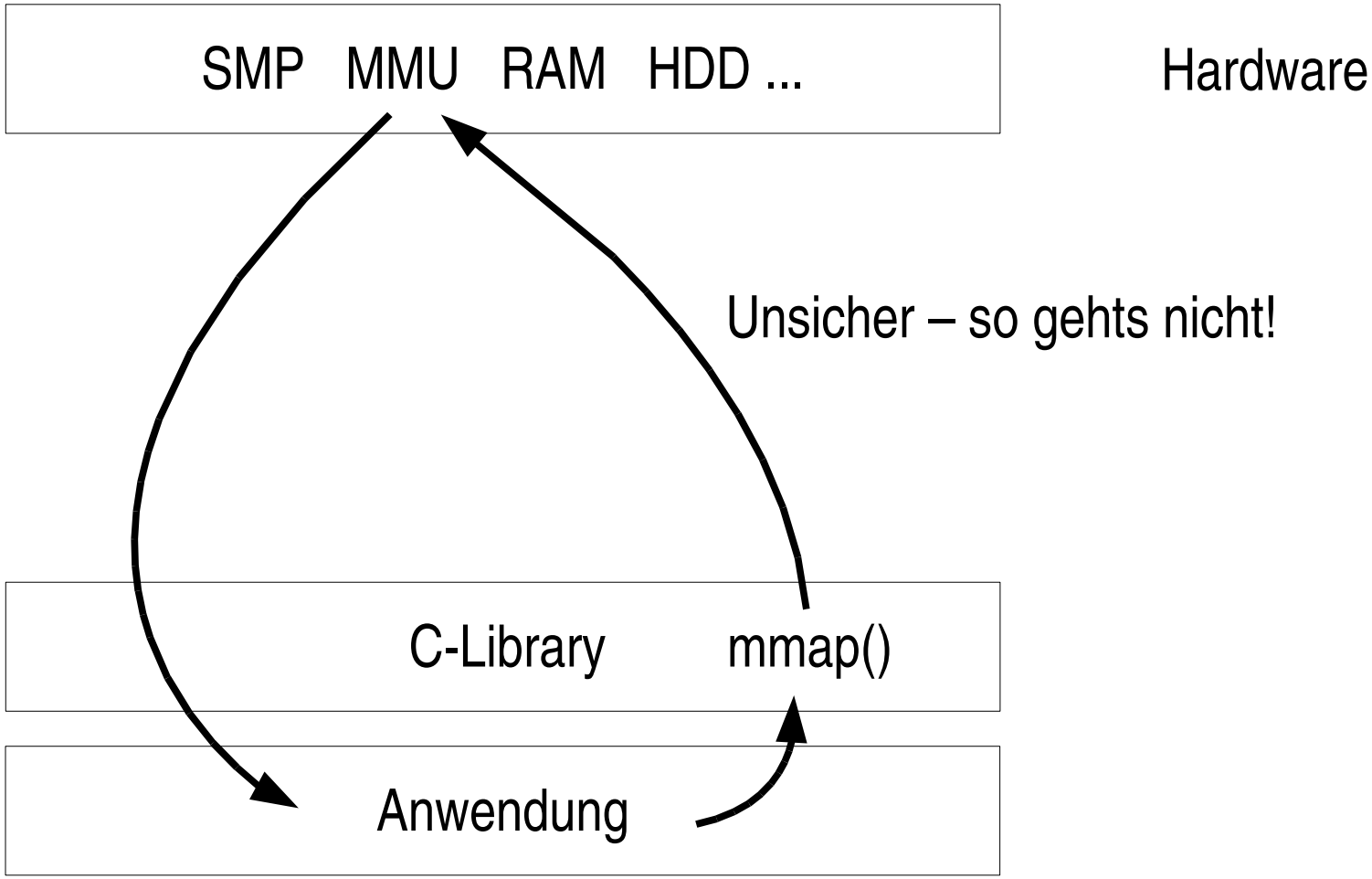
- keine ggs. Performance-Beeineinträchtigung der DomUs

  Forkbombe in einer DomU,

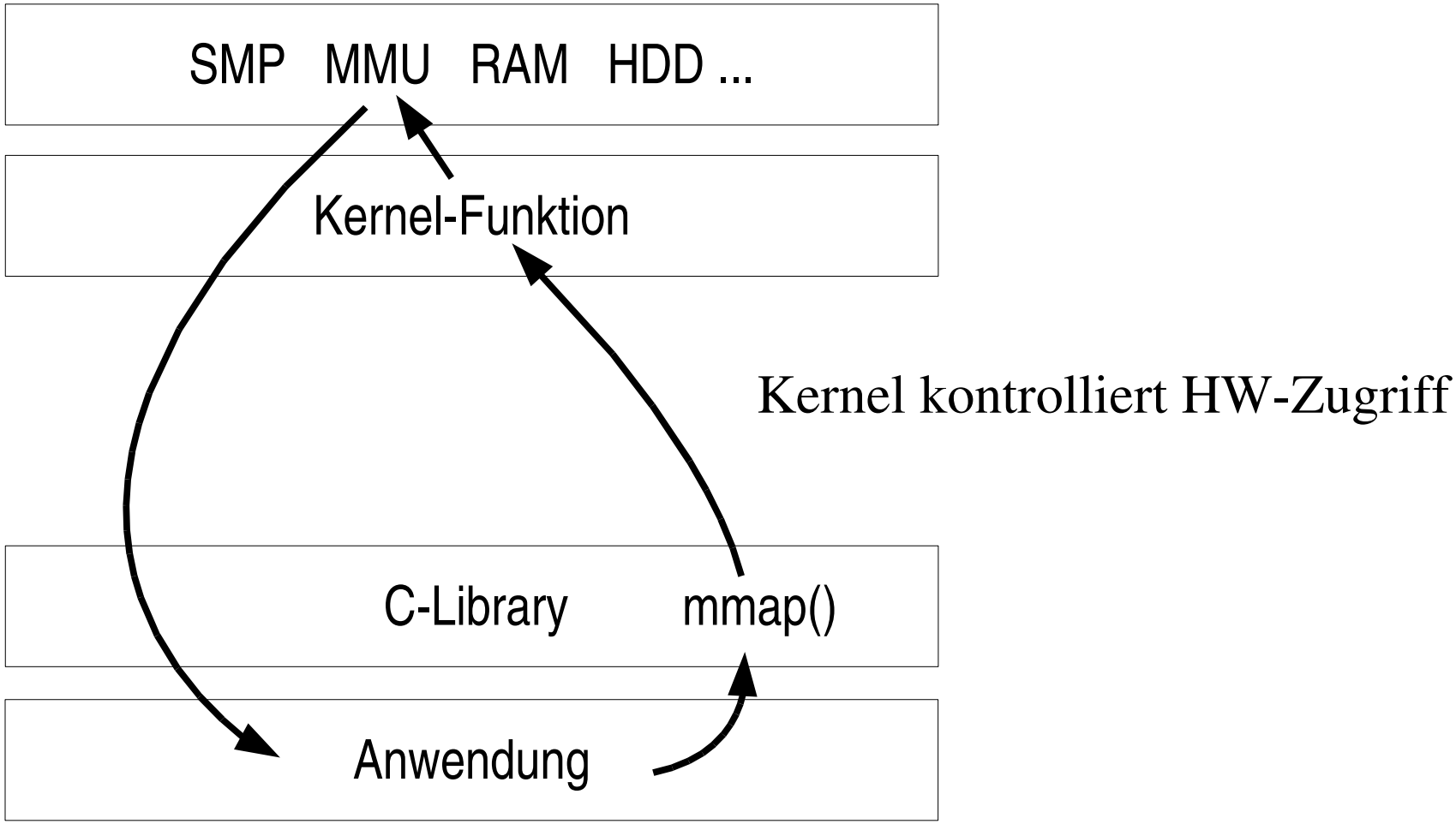
  I/O Trashing, ...

- Heterogene Betriebssysteme / eigene Kernel in den DomUs

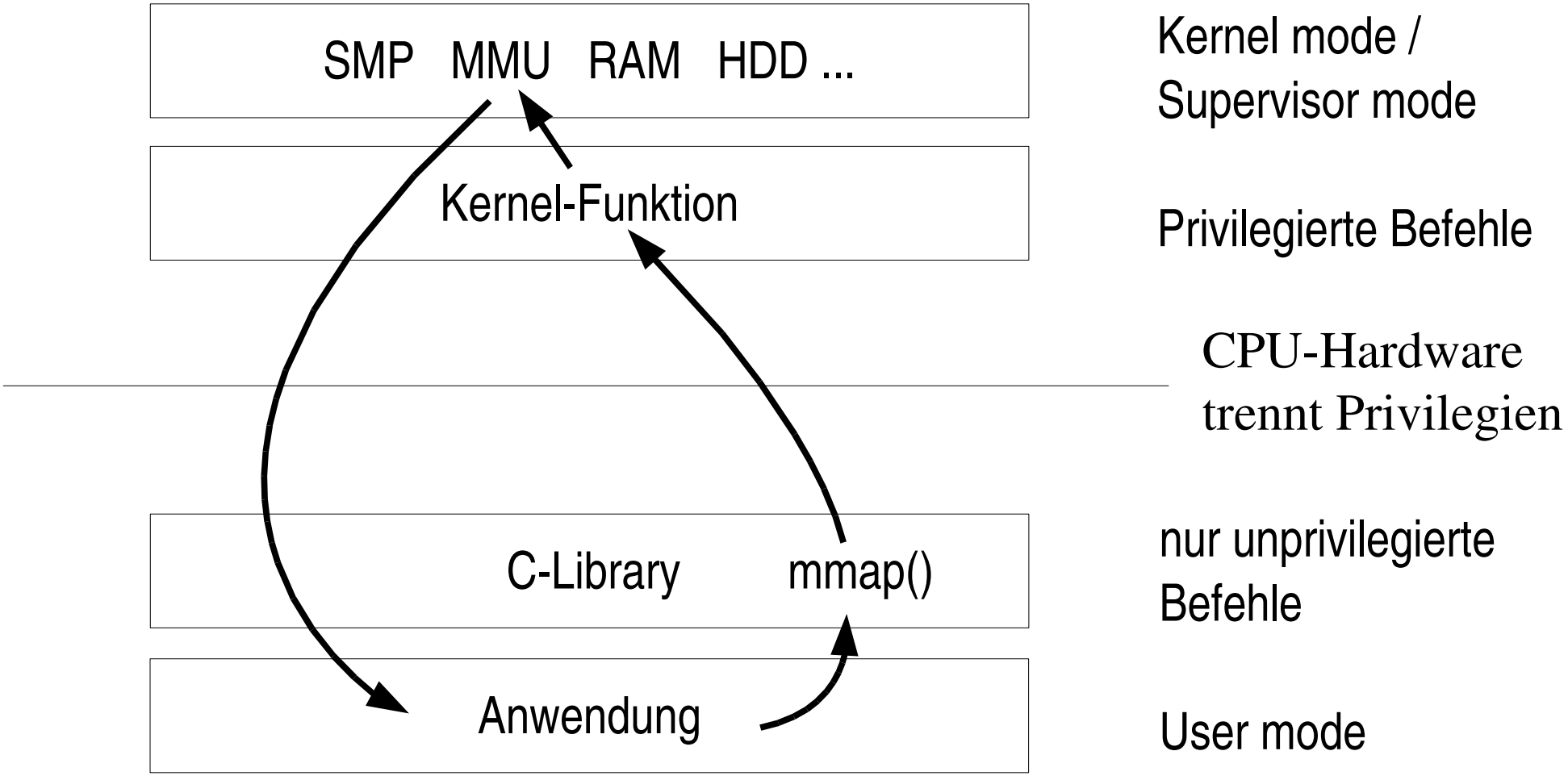
Zugriff auf HW Ressourcen



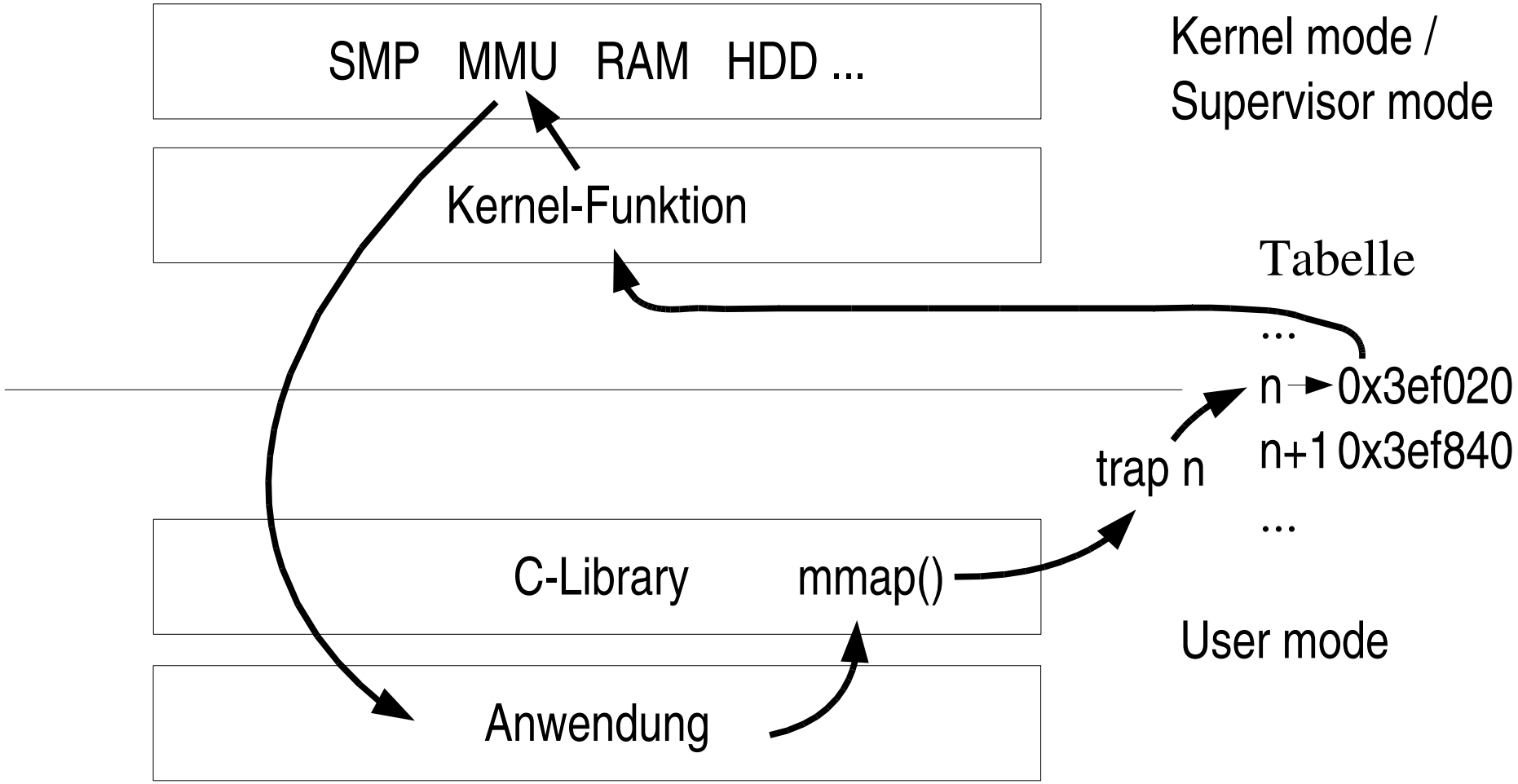
Zugriff über Kernel-Funktion



Direkte Manipulation verhindern

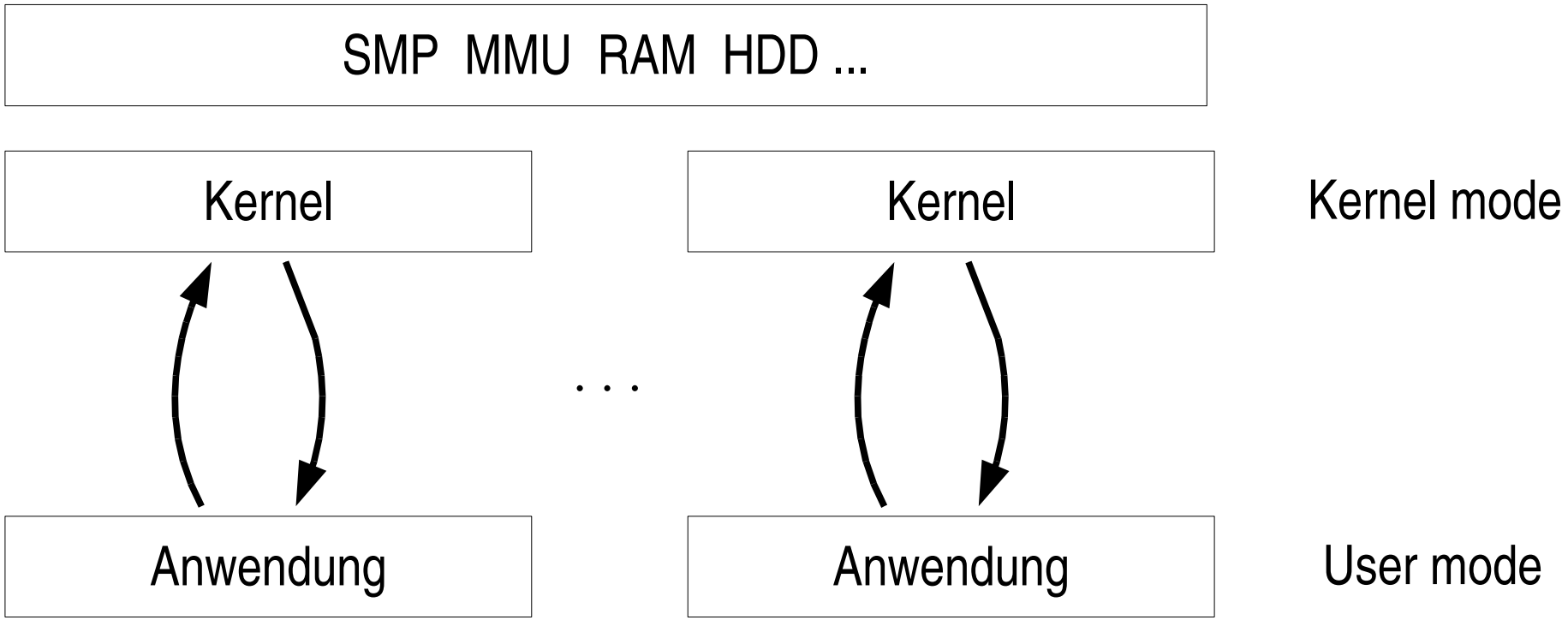


geordnete Eskalation User mode / Kernel mode



Verallgemeinerung auf mehrere BS

VM-Kernel im Kernel-Mode betreiben funktioniert nicht!



Besonderheit x86-Architektur

Ring 0 = Kernel-Mode

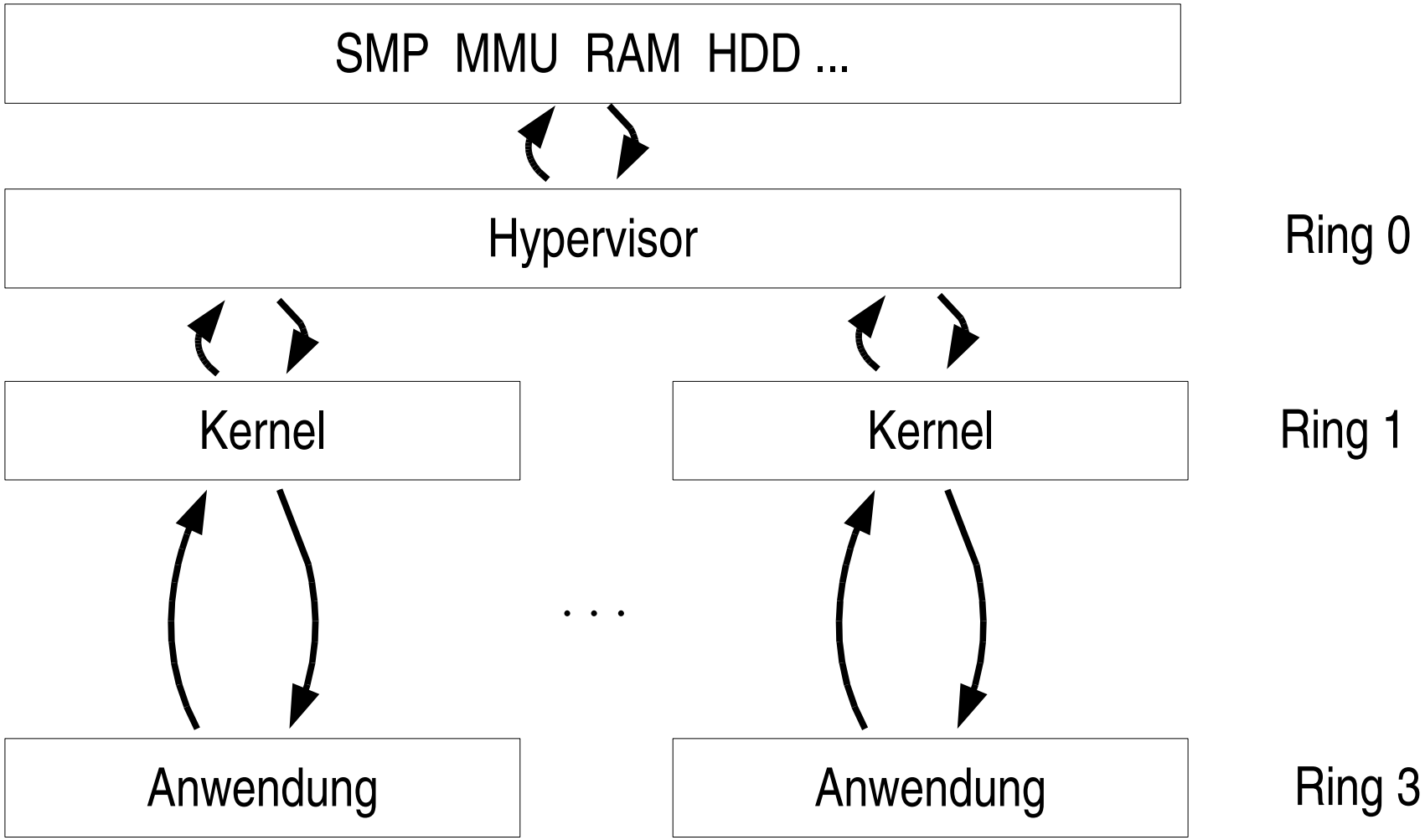
Ring 1

Ring 2

mit Ausnahme von OS/2 ungenutzt

Ring 3 = User-Mode

Hardwarezugriff über Hypervisor



Leistung der Virtualisierung

jedem Gast-Kernel alleinige Kontrolle über HW vorgaukeln

➞ HW geeignet partitionieren und/oder simulieren

(ehemals) privilegierte Befehle der Gast-Kernel

➞ umlenken auf Hypervisor-Calls

Volle Virtualisierung

- sämtliche HW virtualisieren
- privilegierte Befehle im Gast per Trap auf Hypervisor umleiten

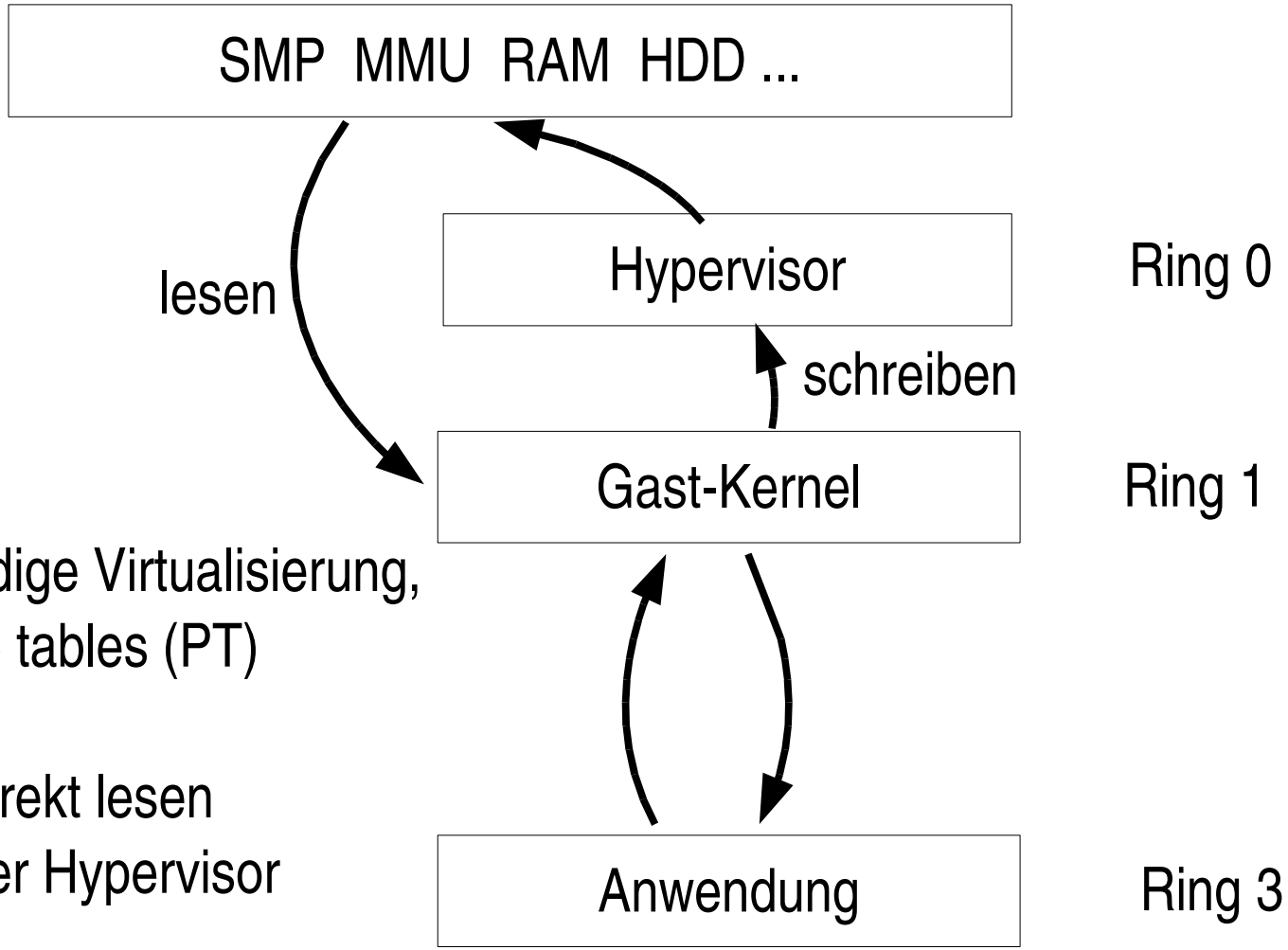
Problem bei x86: einige privilegierte Befehle erzeugen in Ring >0 keinen Fehler

➞ Gast-Kode dynamisch auswerten und umschreiben (vgl. Dtrace)

➞ kostet viel Performance, erlaubt aber unmodifizierte Gast-Kernel

- Beispiel: vmware, Xen HVM

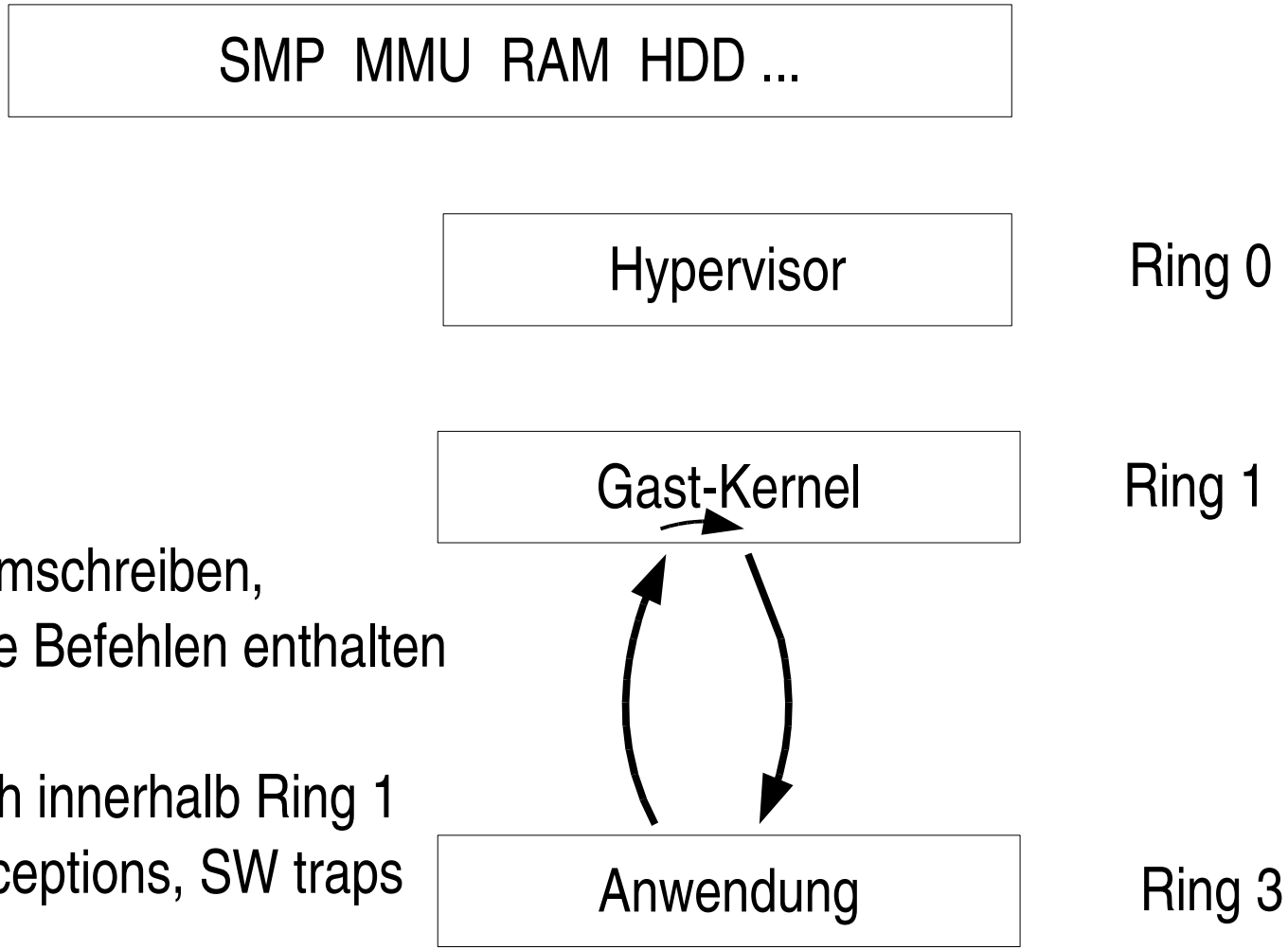
Paravirtualisierung (1)



keine vollständige Virtualisierung,
Beispiel: Page tables (PT)

- eigene PT direkt lesen
- Schreiben per Hypervisor

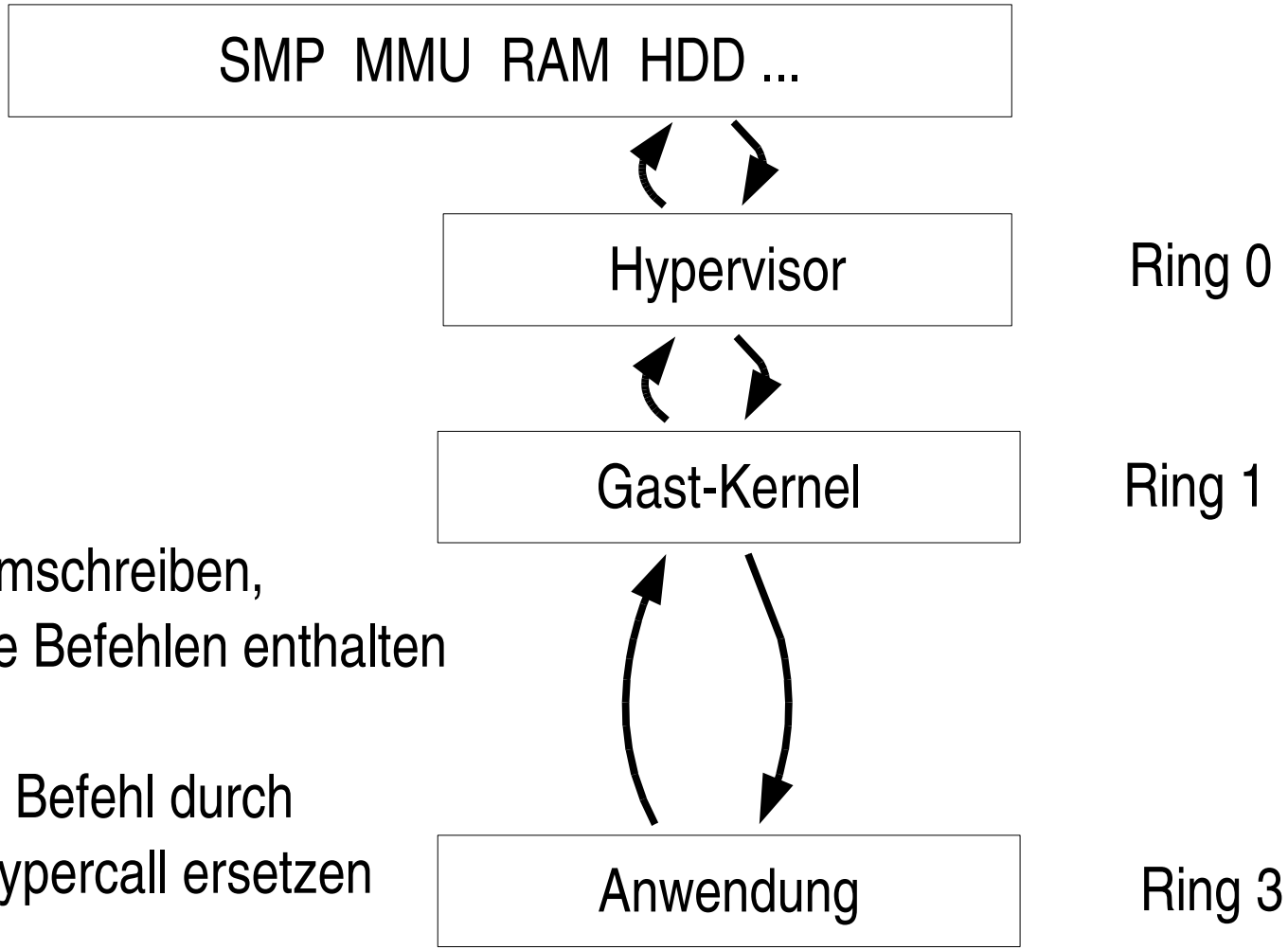
Paravirtualisierung (2)



Kernel-Teile umschreiben,
die privilegierte Befehlen enthalten

- wenn möglich innerhalb Ring 1
  z.B. viele Exceptions, SW traps

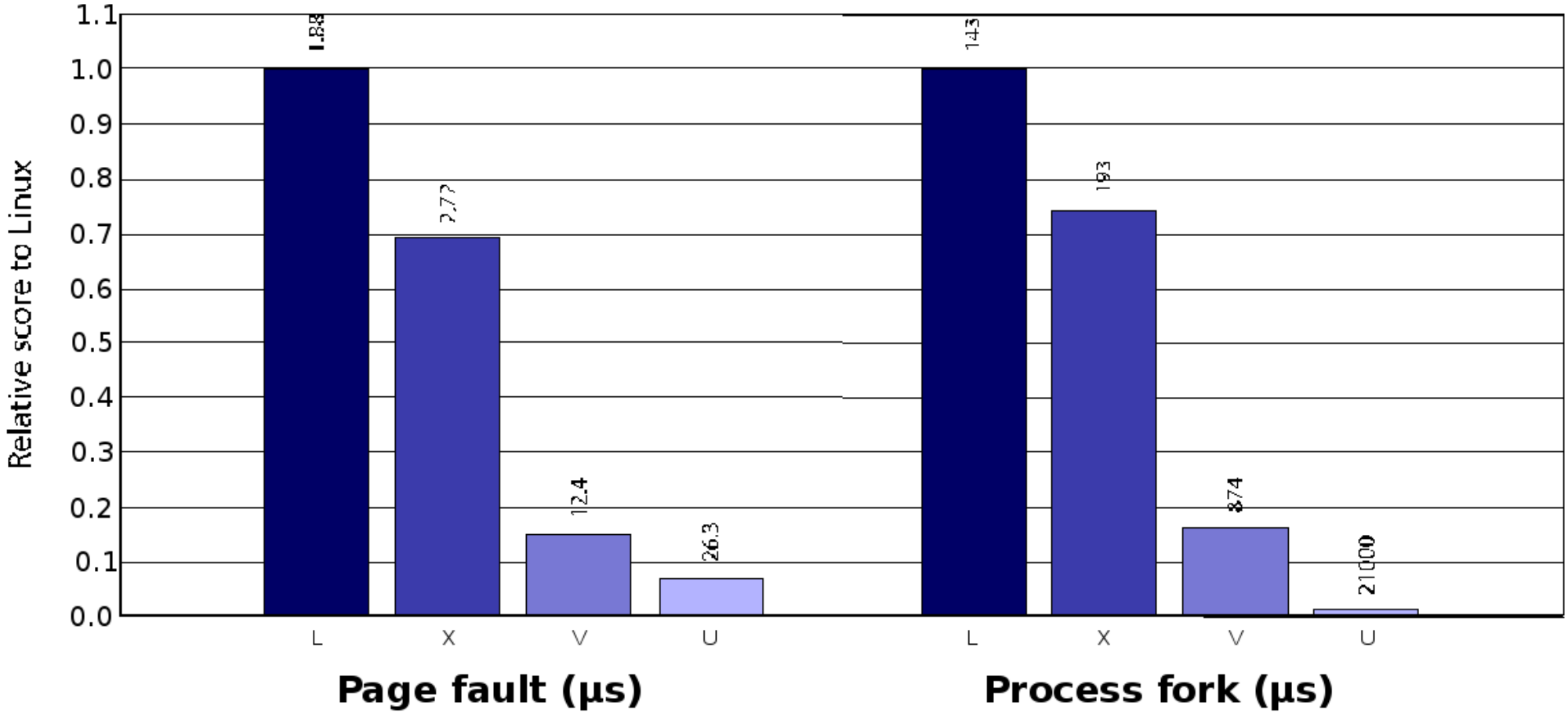
Paravirtualisierung (3)



Kernel-Teile umschreiben,
die privilegierte Befehlen enthalten

- privilegierten Befehl durch hardcoded Hypercall ersetzen

MMU-Microbenchmarks



Linux (L), Xen (X), VMWare Workstation (V), User mode Linux (U)

Quelle: Ian Pratt, http://www.cl.cam.ac.uk/netos/papers/2005-xen-may.ppt

Was wird wo virtualisiert?

Hypervisor virtualisiert selbst:

- Speicherverwaltung
- Scheduling

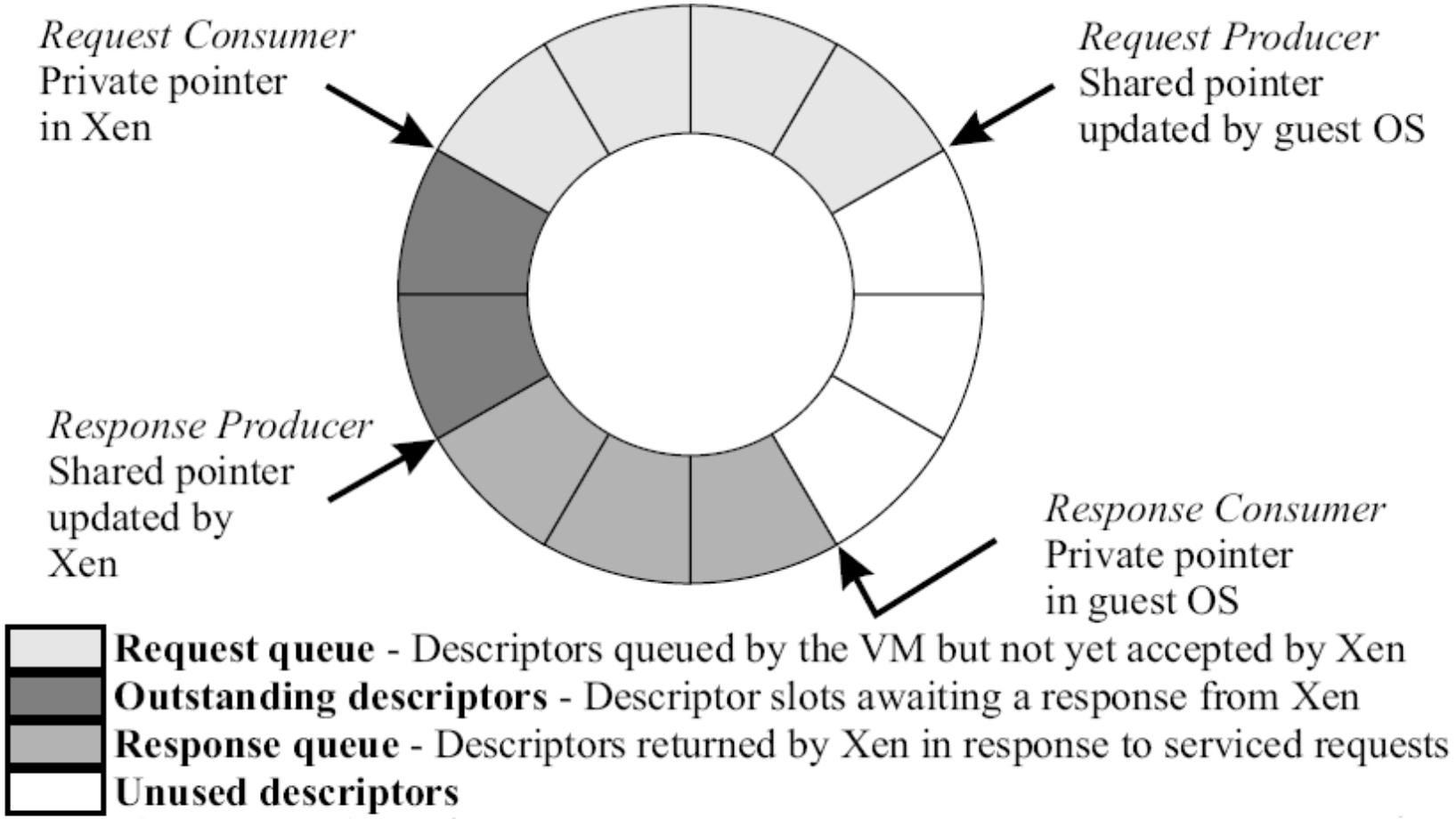
Hypervisor liefert Framework für

- Interrupts ➞ virtuelle Events
- I/O-Hardware ➞ nur abstrakte I/O-Schicht
  asynchroner Ringpuffer

Dom0:

- liefert die tatsächlichen I/O-Gerätetreiber

I/O-Ringe



Quelle: http://www.cl.cam.ac.uk/netos/papers/2003-xensosp.pdf

Praktischer Teil

Praktischer Teil

mit Xen 3.1.0

Hinweis:

Hypervisor + Xen-Userland nicht kompatibel mit Xen 3.0.x

Xen 3.0.x-Kernel laufen aber unter Xen 3.1.0

Xen-Dom0 installieren (1)

Xen-Dom0 ist

- eine ganz “normale” Linux-Installation
- plus Xen-Tools und Xen-Kernel-Image

Distributionsunabhängige Tarbälle:

http://www.xensource.com/download/index_oss.html

Debian-Pakete:

xen-linux-system-2.6.18-4-xen-amd64

xen-tools ➞ optional, machen das Leben leichter!

Xen-Dom0 installieren (2)

1. Bestehende Linux-Installation nehmen oder eine erzeugen
2. Xen-Userland und -Kernel installieren
   - Tarball: auspacken, cd $(where)/dist, ./install.sh
   - Debian: apt-get ...
3. Hypervisor und Xen-Kernel in /boot/grub/menu.list eintragen
4. Rebooten
5. xm top zur Kontrolle (als root)

Grub-Konfiguration

Normalen Kernel booten:

```
title     Debian GNU/Linux ...
root      (hd0,0)
kernel    /vmlinuz-2.6.18-4-amd64
          root=/dev/sda1 ro
initrd    /initrd.img-2.6.18-4-amd64
```

Dom0 booten:

```
title     Xen Dom0 Debian GNU/Linux ...
root      (hd0,0)
kernel    /xen-3.0.3-1-amd64.gz
module    /vmlinuz-2.6.18-4-xen-amd64
          root=/dev/sda1 ro
module    /initrd.img-2.6.18-4-xen-amd64
```

Xen-Userland

xm	Xen monitor, Kommandozeilen-Schnittstelle
xend	verwaltet die virtuellen Domains
xenstore	verwaltet Änderungen in der “Live”-Konfiguration
xenbus	Kommunikation zwischen den virtuellen Domains
/etc/xen/*	Konfigurationsdateien

Xen-Tools

Grundkonfiguration in /etc/xen-tools/xen-tools.conf

➞bekommen alle neu angelegten DomUs

```
lvm = single          # Volume group “single”; man will LVM benutzen!
image  = full         # Specify sparse vs. full disk images. LVM implies “full”
size   = 2Gb          # Disk image size.
memory = 512Mb        # Memory size
swap   = 512Mb        # Swap size
kernel = /boot/vmlinuz-2.6.18-4-xen-vserver-amd64
initrd = /boot/initrd.img-2.6.18-4-xen-vserver-amd64
arch=amd64            # We want to install a 64bit distro
debootstrap = 1       # Bootstrap for our distro (Debian in our case)
dist   = etch         # Default distribution to install (we want Debian Etch)
dhcp = 1              # DomU is going to boot using DHCP
passwd = 1            # Change root password during setup
```

Grundeinstellung für Netzwerk prüfen

in /etc/xen/xend-config.sxp:

(network-script network-bridge)
#(network-script network-dummy)

➞alle DomU erhalten Bridge auf erstes Interface der Dom0

Auch komplexere Topologien konfigurierbar:

- verschiedene DomUs auf verschiedene Interfaces bridgen
- Dom0 kann zwischen DomU und Außenwelt routen
- DomU werden über NAT angebunden

Xen-Tools: Neue DomU erzeugen

xen-create-image --hostname grag [--ip 129.70.142.241]

tail -f /var/log/xen-tools/grag.log

➡ führt Debian-bootstrap über Netz durch,
erzeugt initiale Konfiguration

Xen-Tools: Übersicht

xen-create-image : Neues Abbild erzeugen

xen-delete-image : Abbild wieder löschen

xen-list-images: : Abbilder anzeigen

bootfähige Konfiguration liegt in /etc/xen

vi /etc/xen/grag.cfg

für DHCP-Boot brauchen wir eine Mac-Adresse:

vif = ['']

“Default-Bein”

vif = ['mac=00:16:3E:2C:F0:01, bridge=xenbr0']

Weitere Interfaces zur Verfügung stellen:

/etc/xen/scripts/network-bridge start vifnum=1 bridge=xenbr1 netdev=eth1

DomU booten

Erster Boot am besten mit Konsole (-g später weglassen):

xm create -c grag.cfg

Konsole abgeben: Ctrl-5

Konsole holen: xm console grag

Kontrolle und erste Hilfe

```
xm top              # laufende Domains anschauen
xm list [-l]        # mehr Infos zu laufenden Domains

xm shutdown grag    # eine DomU anhalten (= shutdown -h)
xm destroy grag     # eine DomU abbrechen (= Stecker ziehen)
```

Statische Speicherverwaltung

Grundeinstellung:

- dom0 hat allen freien Speicher
- domU erhält daraus festes Kontingent beim Erzeugen

➞ dynamische Relokation über balloon-Treiber möglich

Dynamische Speicherverwaltung (1)

1. Speicher der dom0 begrenzen:

/boot/grub/menu.lst:

```
...
kernel        /boot/xen-3.0.3-1-amd64.gz dom0_mem=192M
...
```

Dynamische Speicherverwaltung (2)

2. größere Seitentabellen für DomU-Kernel:

/etc/xen/grag.cfg:

```
...
memory   = '512'
maxmem   = '2048'
extra    = 'mem=2048M'
```

(ab Xen 3.1.0 entbehrlich)

➞ DomU startet nach wie vor mit 512M,
kann aber auf bis zu 2G vergrößert werden

Dynamische Speicherverwaltung (3)

Speichermenge einer domU verändern:

```
root@dom0# xm mem-set grag 376
```

Speichermenge aus der domU heraus verändern:

```
root@grag# echo $((700*1024*1024)) >/proc/xen/balloon
```

Dynamische Speicherverwaltung - Ausblick

- momentan nur manuelle Relokation
- in Zukunft paravirtualisierter out-of-memory-Treiber

Allerdings: grundsätzlich kein Überbuchen möglich!

domU anhalten

xm pause curtis

xm unpause curtis

domU speichern und starten

cd /home/xen/domains/curtis
xm save curtis curtis.xen

... Maschine rebooten oder was auch immer ...

xm restore curtis

Achtung: TCP timeout bei ssh, X11

➞ screen oder VNC verwenden

domU migrieren (1)

xm migrate curtis dom0.dotsero.techfak.uni-bielefeld.de

Stolperfalle:
vom “flipping” eth-Driver auf “copying” eth-Driver umschalten

/etc/xen/curtis.cfg:

```
...
extra   = 'xennet.rx_copy'
...
```

domU migrieren (2)

Momentan: lokaler Storage wird (noch) nicht übertragen

Empfohlen: Root-Filesystem per NFS, SAN, iSCSI,...

Workaround: Dateibasierte Abbilder verwenden

```
dom0-alt> xm save curtis
dom0-alt> # per scp Abbild(er) auf neue dom0 übertragen
dom0-neu> xm restore curtis
```

Vorsicht: TCP timeout

HVM - Hardware-Virtualisierung

Benötigt:

- Vanderpool (Intel, vmx) oder Pacifica (AMD, svm):

  xm dmesg | grep HVM

- LVM-Volume für HD-Emulation
- Iso-Abbild des Installations-Datenträgers
- SDL oder VNC für graphische Konsole des Gastes

HVM - Konfiguration

- /etc/xen/xmexample.hvm entsprechend anpassen, u.a.:

```
kernel = "/usr/lib/xen/boot/hvmloader"
builder='hvm'
disk=[  'phy:/dev/stripe/joan-disk,ioemu:hda,w',
        'file:/home/xen/domains/joan/winxp.iso,hdc:cdrom,r']
boot="d"  # Nach Installation auf “c”setzen
sdl=0
vnc=1
```

- vncpasswd + Paßwort in /etc/xen/xend-config.sxp eintragen

HVM-Gast booten

Booten/Verwalten wie gewohnt:

xm create joan

- Umstellen auf boot='c' nach Installation nicht vergessen!

- Unterstützte OS:
  Windows XP, ... (Stille, tödliches Schweigen ;-) ...

Zusammenarbeit mit Intrusion Detection

Intrusion Detection-Systeme:

- erstellen Datenbank mit Prüfsummen aller Dateien/Directories
- daran werden Einbrüche in das System erkannt

Prinzipbedingte Schwachstelle:

- IDS-Binary und -Datenbank im überwachten System sichtbar und im schlimmsten Fall manipulierbar

IDS mit Xen

Idee:

- IDS aus der Dom0 über die DomU-Gäste laufen lassen

➞ kompromittierte DomU können IDS nicht erkennen/manipulieren

Zugriff auf Dateisystem der DomU - falscher Ansatz:

dom0> mount /dev/stripe/grag-disk /mnt

➞ Dateisystem ist unter zwei Kerneln gleichzeitg gemountet

Zugriff auf DomU über LVM-Snapshots

```
dom0> lvcreate --snapshot -L 1G -n snap stripe/grag-disk
dom0> mount /dev/stripe/snap /mnt
```

... Backup, Intrusion Detection auf /mnt laufen lassen ...

```
dom0> umount /mnt
dom0> lvremove stripe/snap
```

Xen in der RBG (1)

4 von 5 Linux-Servern (X4100/X4200) haben Xen

1. Mailserver
   - mailin (4 Kerne)
   - smarthost (1 Kern dynamisch, idle)

2./3. Netboot-Server
   - NFS/Netboot (1 Kern pinned)
   - Compute (3 Kerne pinned)
   - mailin/smarthost cold-spare

Xen in der RBG (2)

4. Projektserver
   - Projekt-DomU
   - vserver: TAK
   - vserver: Fachschaft
   - vserver: ...
   - Test-DomU

5. Demnächst: SunRay-Server
   - 32bit-DomU unter 64bit-Dom0 geht jetzt

Zusammenfassung

- Administrative Vorteile: dom0 als Serviceprozessor
- Save/Resume und Live-Migration
- Große Maschinen besser auslasten
- Weniger Hitze/Kosten (Eine große Maschine ersetzt viele kleine)
- Verschiedene Betriebssysteme auf einer Maschine

Ausblick

Was noch kommen wird bzw. wünschenswert ist:

- dynamischere Speicherverwaltung

  (momentaner Workaround: vserver innerhalb einer DomU)

- Migration von lokalem Storage
- besseres HVM

Danke fürs Zuhören!

Literatur

Lösungen für spezielle Probleme (Cut&-Paste-Suche nach der Fehlermeldung):

- http://lists.xensource.com/archives/html/; insbesondere
  - http://lists.xensource.com/archives/html/xen-users/ und
  - http://lists.xensource.com/archives/html/xen-devel/

Allgemeiner Überblick, Installation:

- http://www.pug.org/index.php/Xen-Installation

Weiterführende Literatur:

- http://wiki.xensource.com/xenwiki/XenDocs
- http://www.cl.cam.ac.uk/netos/papers/2003-xensosp.pdf
- http://www.cl.cam.ac.uk/netos/papers/2005-xen-may.ppt
- Henning Sprang et al: Xen Virtualisierung unter Linux, Open Source Press, 2007