

Xen!

best Open Systems Day  
Fall 2007

Unterföhring

Marco Kühn  
best Systeme GmbH  
marco.kuehn@best.de



# Agenda

- Einführung und Übersicht
- Architektur
- Konfiguration
- Live Demo

### Warum Virtualisierung ?

Virtualisierung hilft bei Problemen wie:

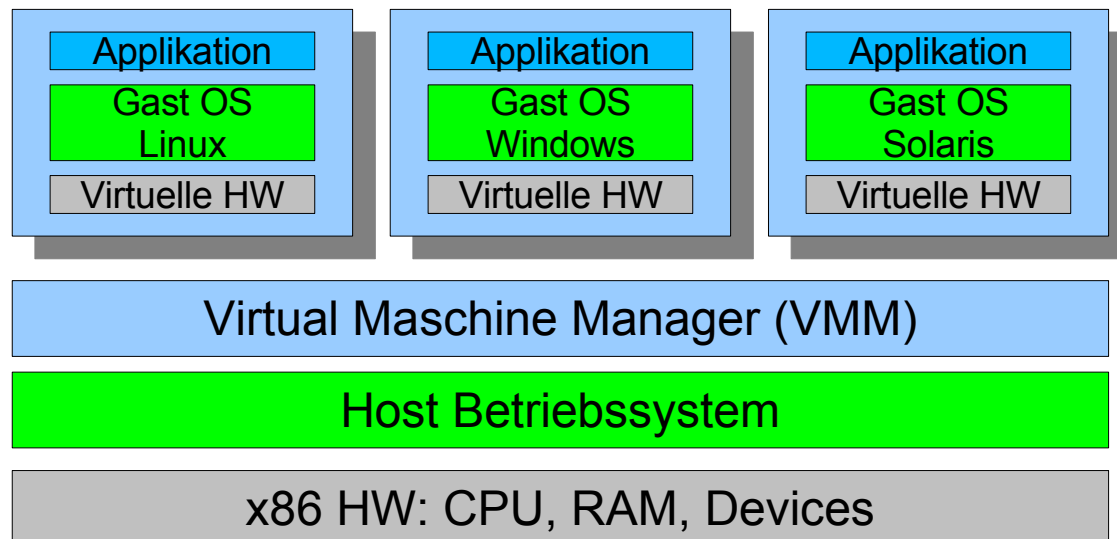
- Hochverfügbarkeit
- Hohen Administrationsaufwand
- Geringe Serverauslastung
- Kosten zu sparen
- Betriebssysteme, die nur mit wenigen Applikationen umgehen können

## Vollständige Virtualisierung (Typ 2 Hypervisor)

Der Virtual Machine Monitor (VMM / Hypervisor) läuft auf einem Host-Betriebssystem als Applikation im Userspace – und fungiert als Übersetzer zwischen Gast und Wirt.

Die Virtualisierung wird vor dem Gast vollständig verborgen.

Bekannte Vertreter sind: VMware (Workstation, GSX), MS Virtual PC / Server



+ kostenfrei

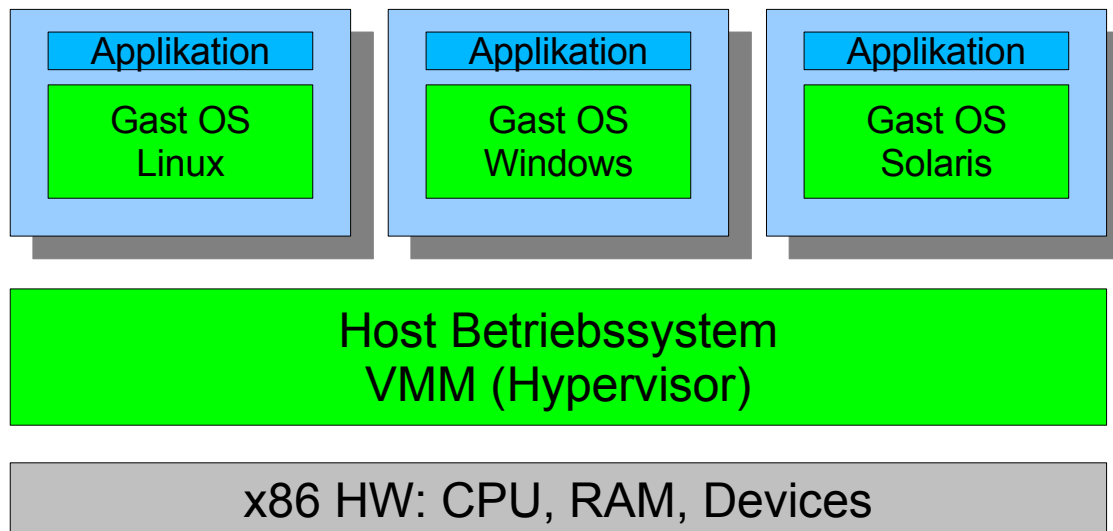
- schlechte Performance
- schlechte Qualität

## Vollständige Virtualisierung (Typ 1 Hypervisor)

Der Virtual Maschine Monitor (VMM / Hypervisor) läuft auf einem Host-Betriebssystem im Kernel – und fungiert als Übersetzer zwischen Gast und Wirt.

Die Virtualisierung wird vor dem Gast vollständig verborgen.

Bekannte Vertreter sind: VMware ESX, Xen, Sun LDomS



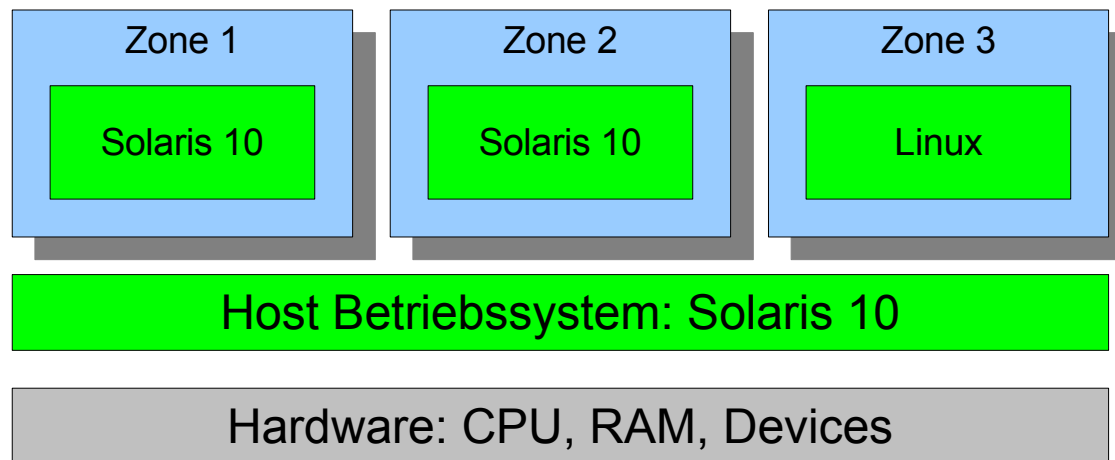
- + geringe Kosten Xen, LDomS
- + gute Performance
- + Qualität

- hohe Kosten ESX
- eingeschränkte HW

## Virtualisierung auf OS Ebene

Mehrere Instanzen ein und desselben Betriebssystems laufen virtuell auf einem Kernel. Man spricht auch von Single Kernel Image (SKI). Das Betriebssystem erzeugt mehrere Instanzen, die als „Container“ verwaltet werden. Den Containern werden Ressourcen zugewiesen.

Bekannte Vertreter: Virtuozzo und Solaris Zones



- + geringe Kosten
- + sehr gute Performance
- + Device Management
- + Resource Management

- OS Release



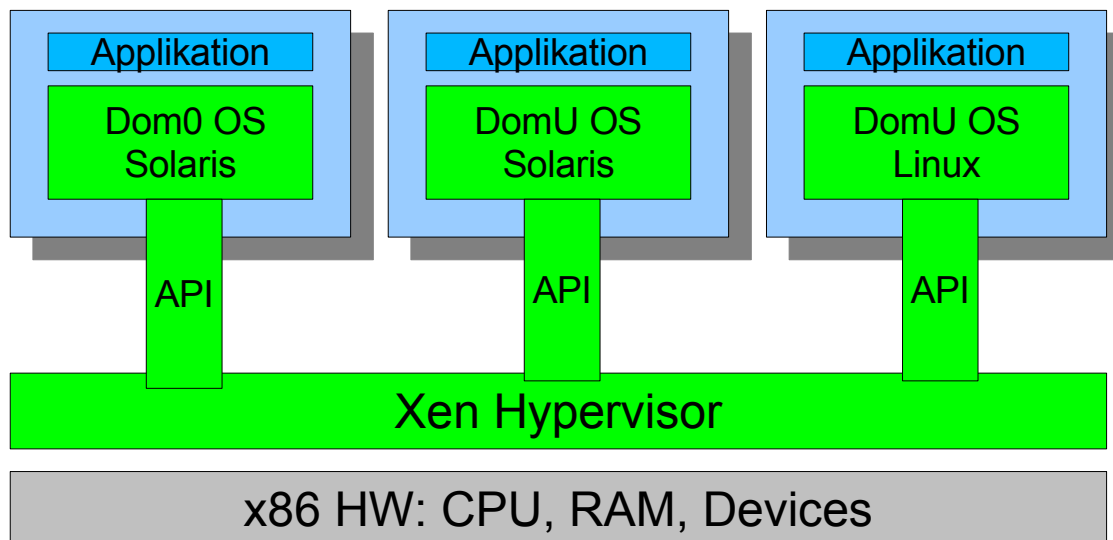
## Para-Virtualisierung

Voneinander unabhängige virtuelle Maschinen auf Basis ihres eigenen Betriebssystems greifen über eine bereitgestellte Schnittstelle direkt auf die Hardware zu – gesteuert und kontrolliert durch den Hypervisor.

Die Gäste wissen, daß sie virtuelle Maschinen sind und mit dem Hypervisor kommunizieren (Hypercall).

Das erfordert die Anpassung des Gastbetriebssystems (Kernel oder PV Treiber).

Bekannte Vertreter: Xen



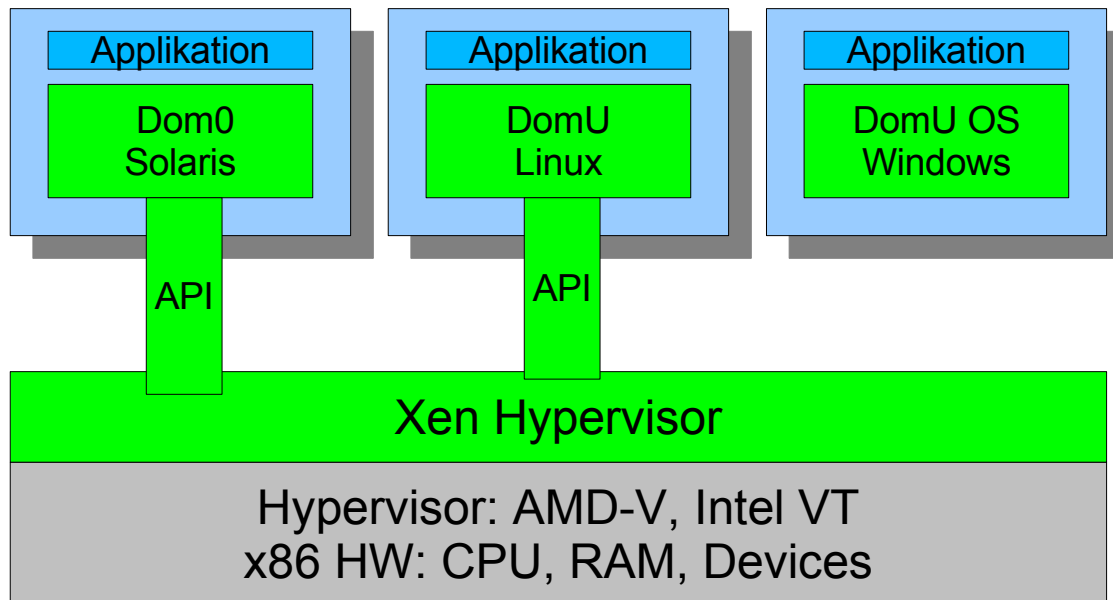
- + geringe Kosten
- + gute Performance
- + Qualität

- OS Windows

## Hardwareunterstützte Virtualisierung

AMD und Intel bieten x86 Prozessoren mit Hypervisor an für folgende Virtualisierungskomponenten:

- CPU
- RAM
- I/O (auf Basis IOMMU = I/O Memory Management Unit)
- Tagged TLB (Translation Lookaside Buffer) (AMD)



## VMware ESX:

- Hardwareunterstützte Virtualisierung
  - Typ 1 Hypervisor
- Vollständige Virtualisierung
- ESX Server ist Hypervisor



## Xen:

- Hardwareunterstützte Virtualisierung
  - Typ 1 Hypervisor
- Paravirtualisierung
- Vollständige Virtualisierung
  - Basis: AMD-V, Intel-VT
- Xen Kernel <50k Zeilen ca. 0.25 MB



Windows Longhorn  
mit Hypervisor

# Agenda

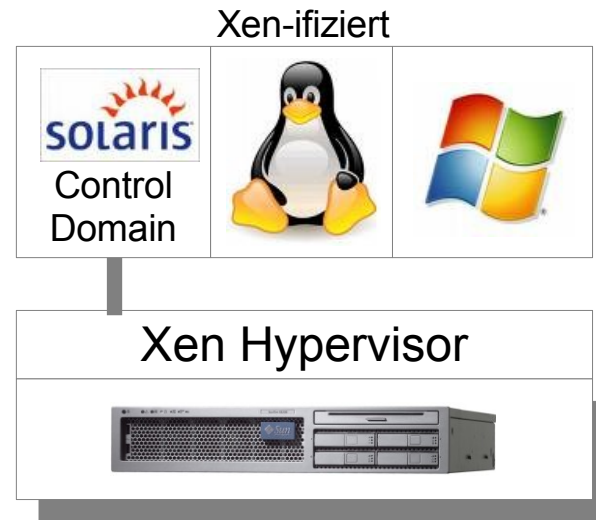
- Einführung und Übersicht
- **Architektur**
- Konfiguration
- Live Demo

Xen kennt zwei Operationsmodi:

**Paravirtualisierung**, die die Anpassung am Gast-Kernel oder PV Treiber voraussetzt

**Vollständige Virtualisierung** („Full Virtualization“), welche eine bestimmte Hardware voraussetzt, aber keine Änderung mehr am Gast

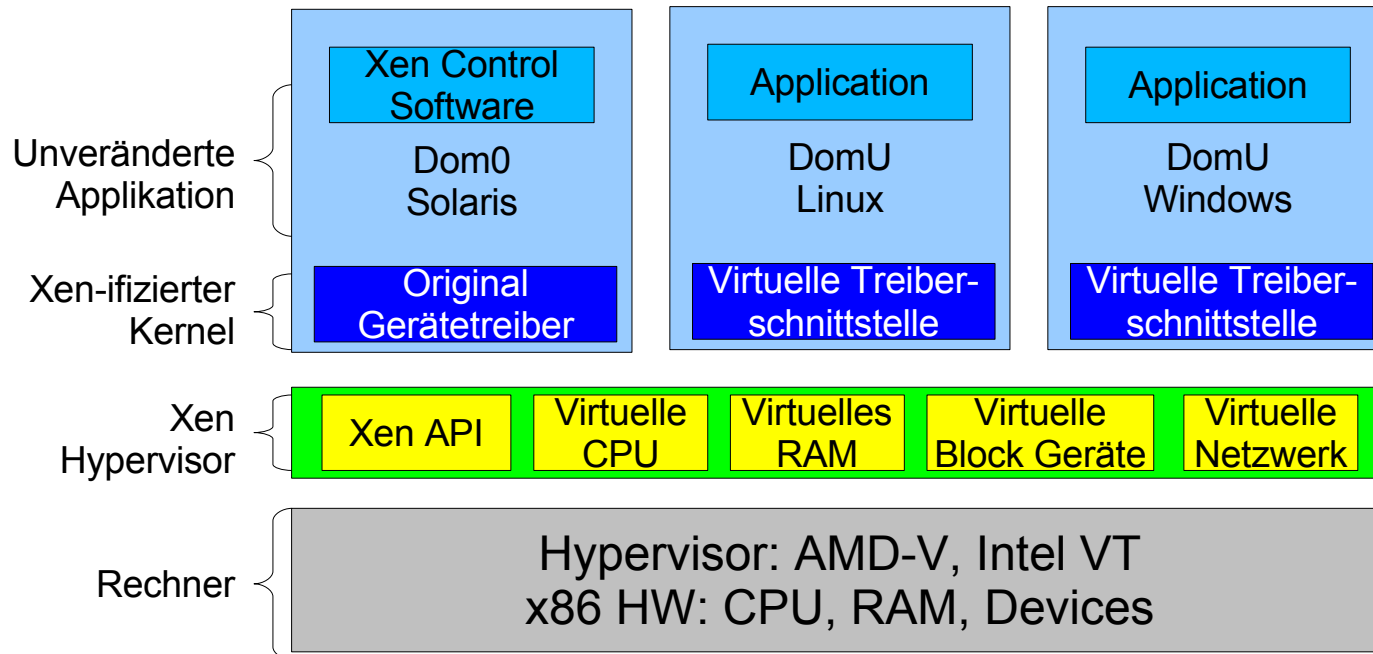
Beide Ansätze sind verschieden, lassen aber einen gemischten Betrieb zu.



# Xen-Architektur

## Systemaufbau

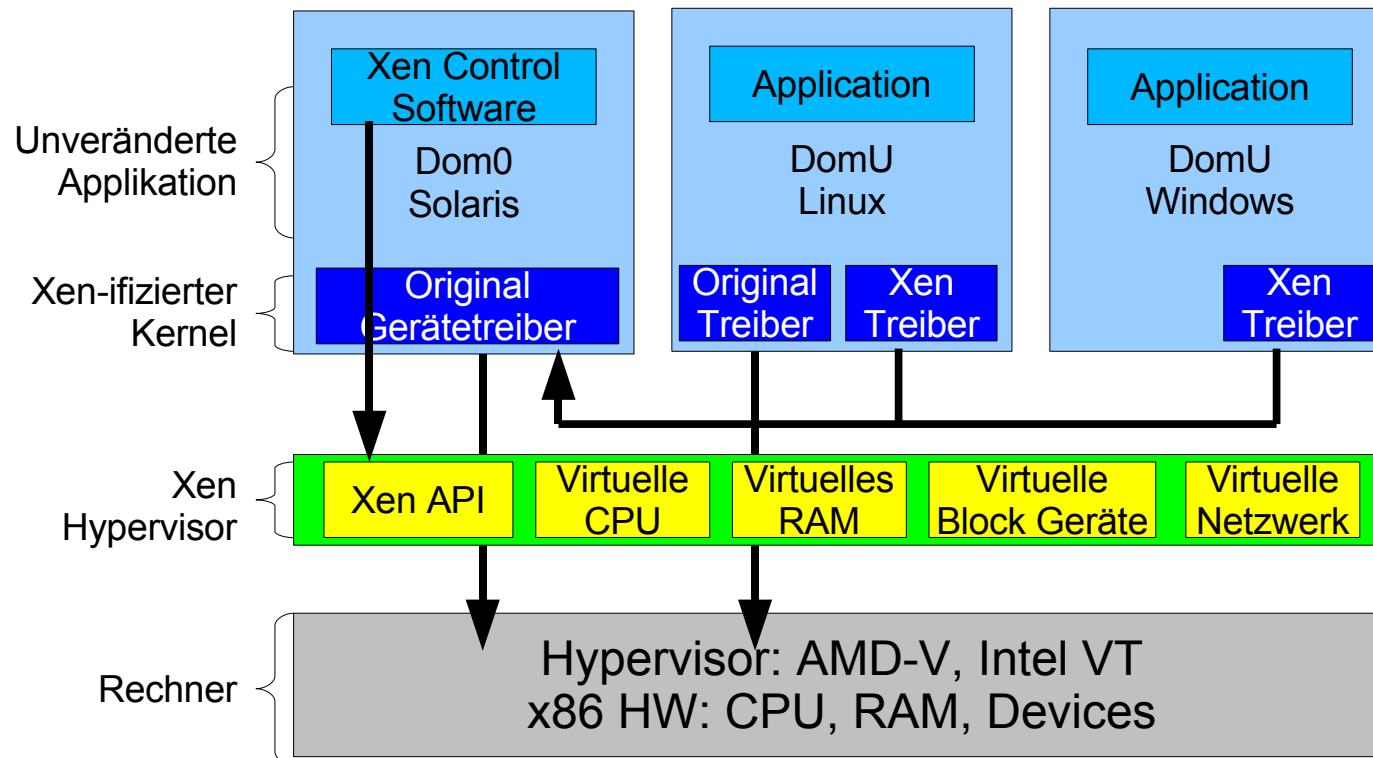
- unterste Schicht, direkt oberhalb der Hardware, ist der Xen-Hypervisor, der beim Bootvorgang als erster geladen wird
- Oberhalb des Xen-Hypervisors sind die Xen-Domains. Die Domain 0 (dom0), auch Control Domain genannt, wird automatisch als Erstes gebootet
- Geräte Zugriffe erfolgen ausschließlich über Domain 0, die die Treiber verfügt
- Die Steuerung der DomU's laufen als Applikation in der Domain 0



# Xen-Architektur

## Xen Treiberarchitektur

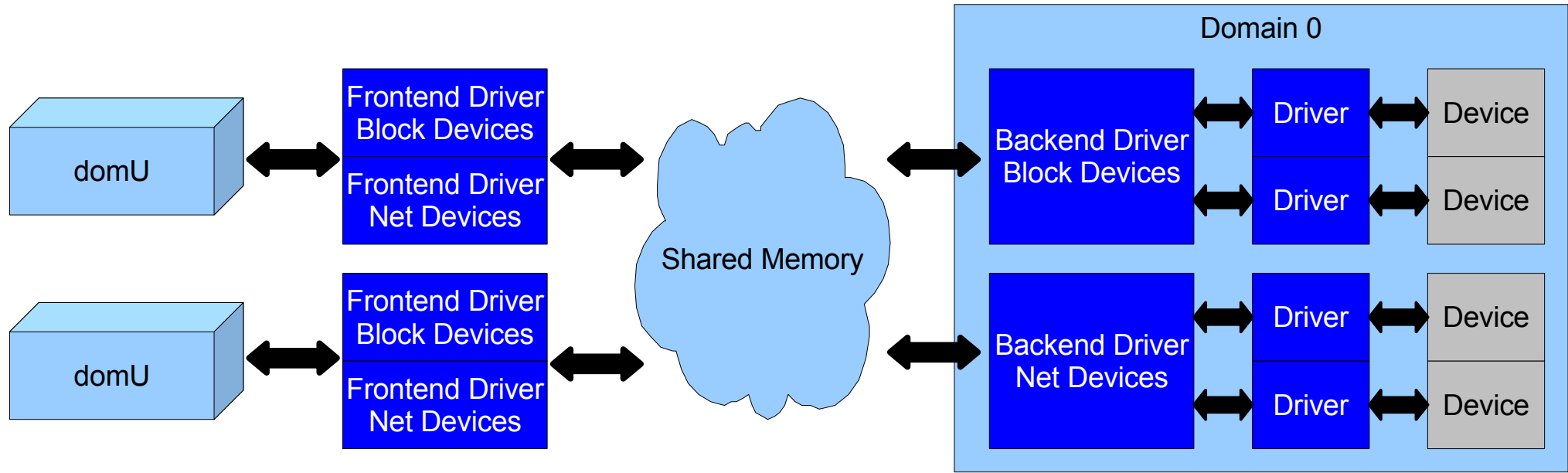
Der Hypervisor als kontrollierende Instanz muss das korrekte Verhalten als Reaktion auf die Instruktionen des Gastes erreichen. VMware filtert diese aufwendig, während die Paravirtualisierung von Xen das richtige Verhalten durch **Hypercalls** erreicht, statt die fragliche Instruktion direkt auszuführen.



## Xen Treibermodell

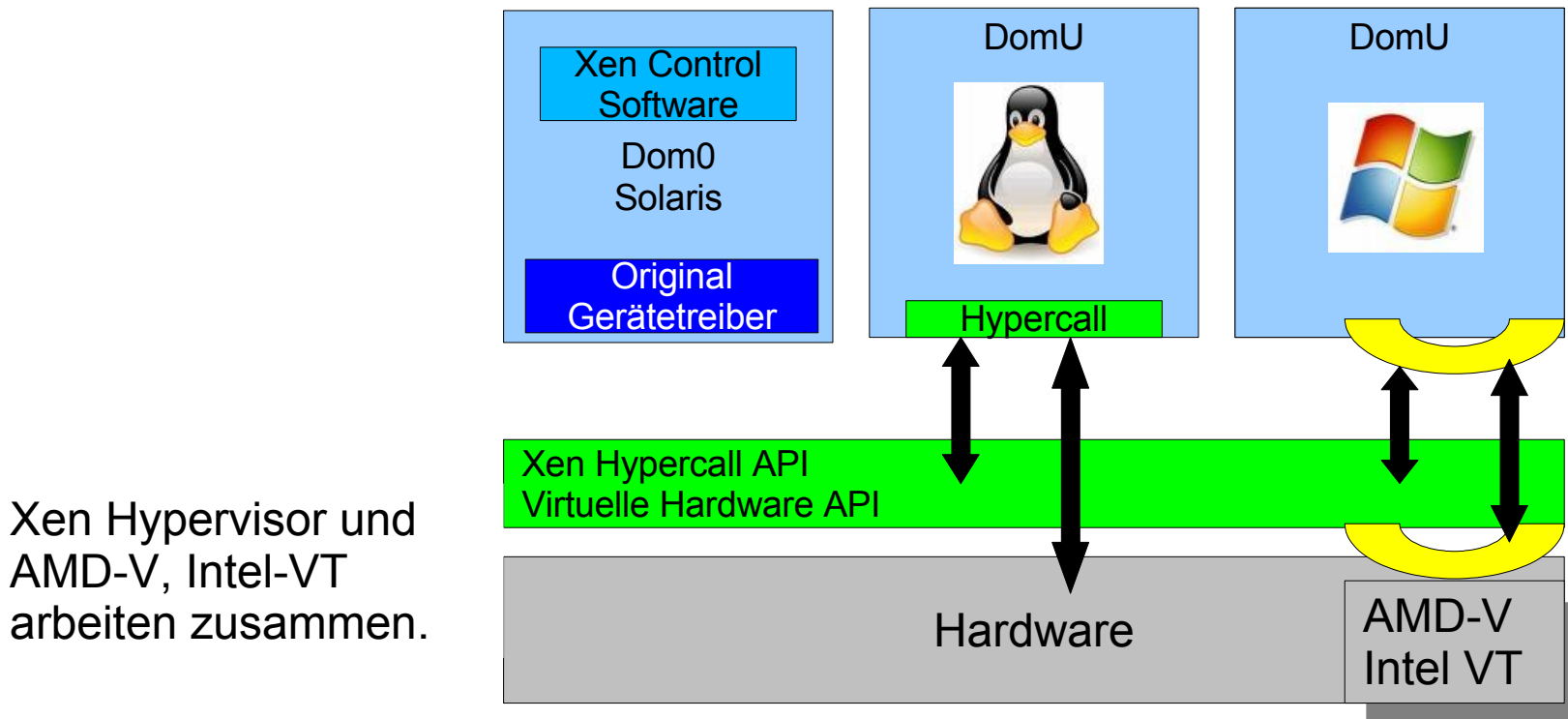
Die Gäste wissen, daß sie virtualisiert sind und benutzen ihren virtuellen Block-Device-Treiber, der nur ein kleiner generischer Frontend-Treiber ist. Die Frontend-Treiber kommunizieren mit den Backend-Treibern der Domain0. Diese kennen die Hardware Adresse des Devices und leiten schlussendlich die Schreib-Lese-Operationen ein.

Diese co-operierende Treiberarchitektur bietet eine hohe Portabilität und eine hohe Ablaufgeschwindigkeit.



## CPU Virtualisierung

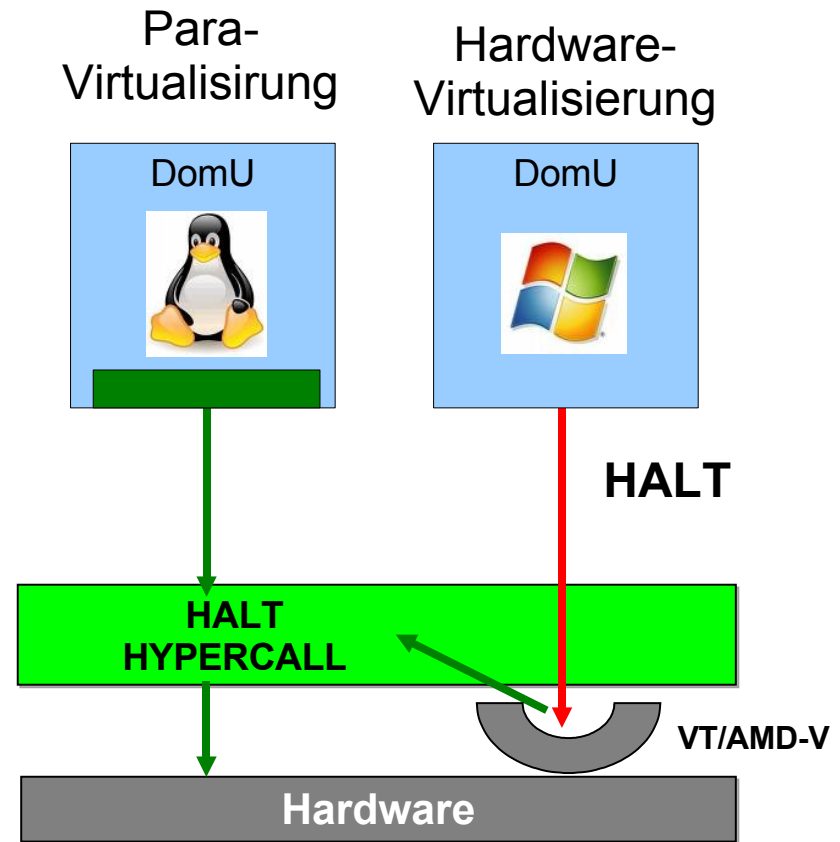
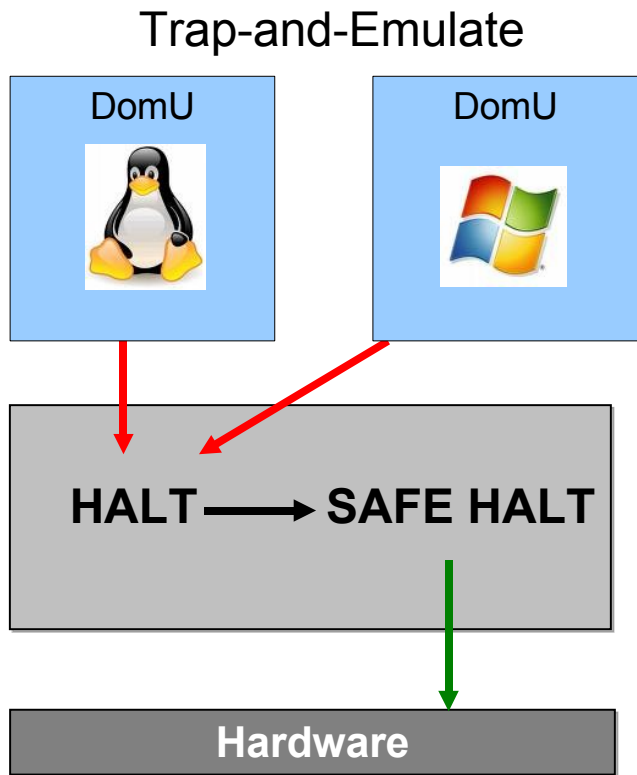
Der Xen Hypervisor bietet eine Xen Hypercall API. Gäste die in CPU Register schreiben wollen oder DMA Operationen durchführen möchten, tun dies über Hypercalls. Nicht paravirtualisierte Gäste wie z.B. Windows sind auf Hardware virtualisierende CPUs wie AMD-V oder Intel VT angewiesen, die unzulässigen Instruktionen abzufangen.



# Xen-Architektur

## CPU Virtualisierung

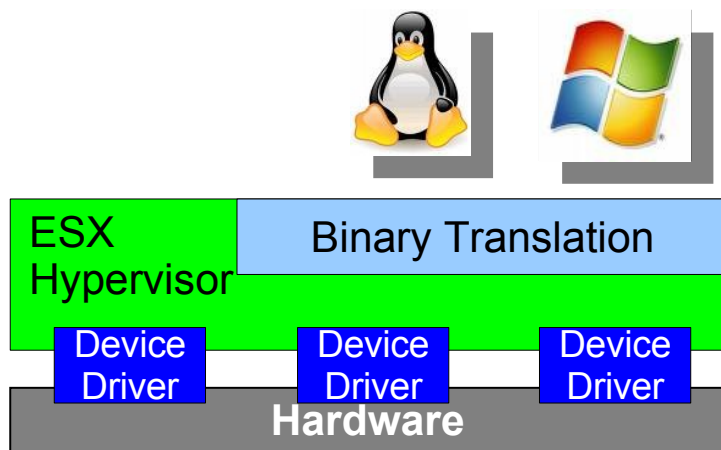
Unterschied bei Xen mit Para-Virtualisierung bzw. HVM im Vergleich zum Binary Patching wie z.B. VMware ESX.



# Xen-Architektur

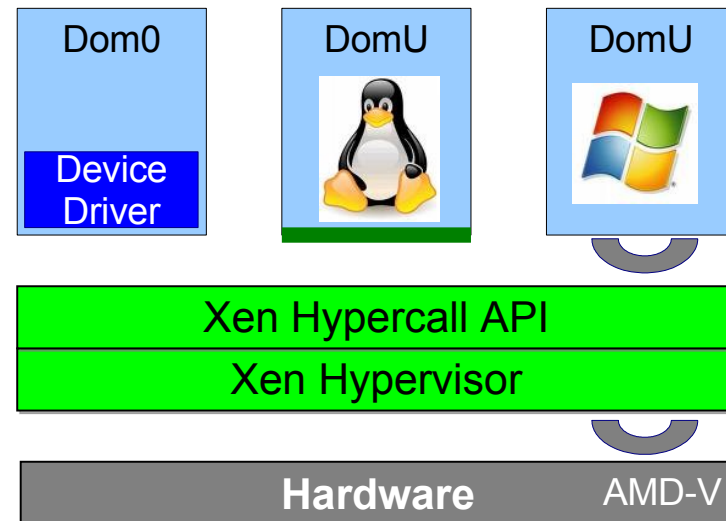
## Virtualisierung in der ersten Generation

- Microkernel unterhalb der Gäste
- Binäres Patchen des OS zur Laufzeit und Geräte Emulation
- Eigene Geräte Treiber
- Performance Overhead



## Para-Virtualisierung

- dünner, effizienter Hypervisor
- Gäste kooperieren mit Hypervisor
- Geräte Treiber liegen außerhalb des Hypervisors
- Minimaler Overhead bei PV Treibern



# Agenda

- Einführung und Übersicht
- Architektur
- **Konfiguration**
- Live Demo

### Features:

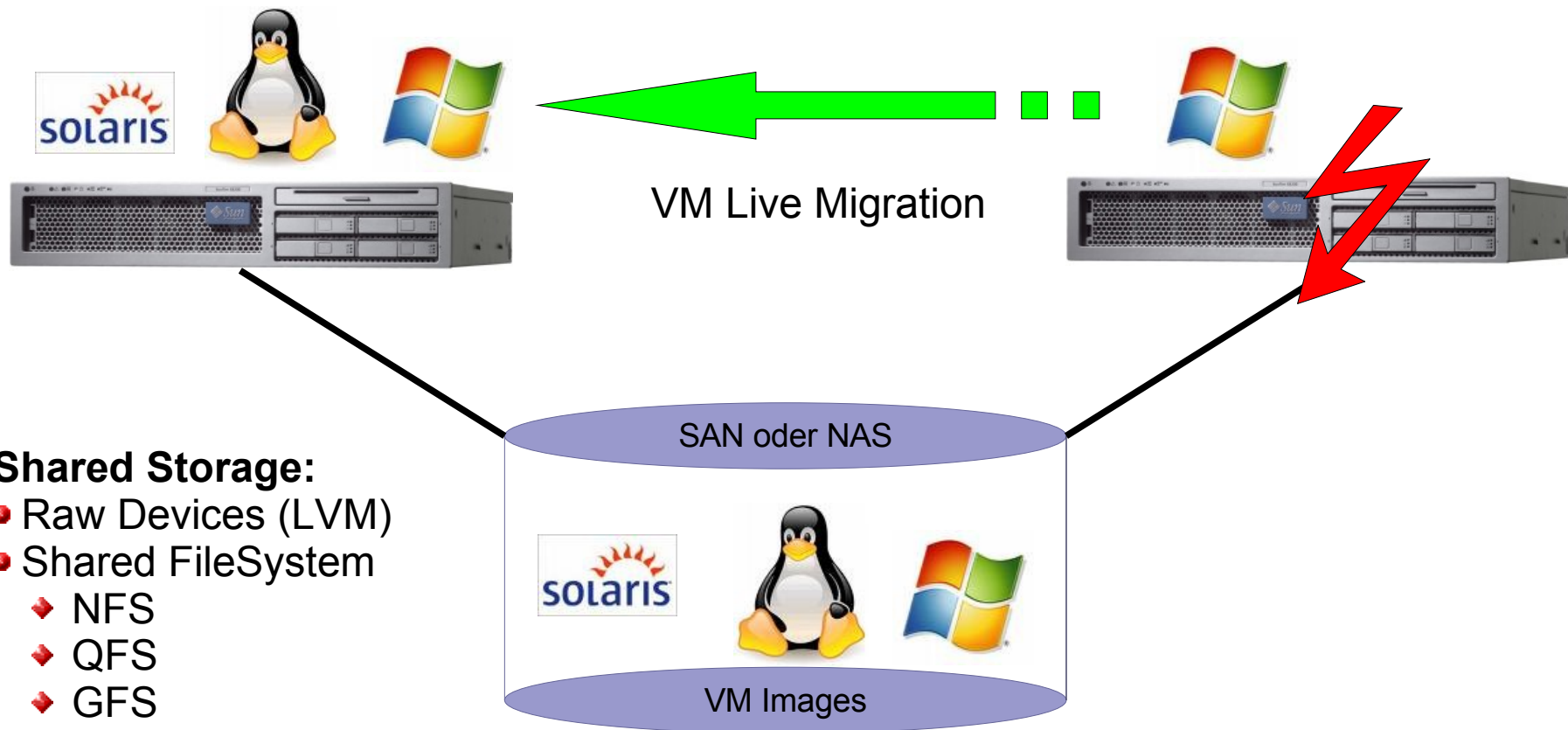
- OS Platform: Suse, RH, Debian, Ubuntu, OpenSolaris, (Solaris)
- 32- und 64-bit Support mit bis 1TB Arbeitsspeicher
- SMP Systeme mit bis zu 32 realen CPUs
- Virtuelle Mehrfach vCPUs auch auf Ein-Prozessor-Maschinen
- Live Migration
- Hot Plugging von Geräten, vCPUs oder Arbeitsspeicher
- Resource Management
- Load Balancing
- Storage Optionen: SAN, NAS, iSCSI, (IB)
- Netzwerke: Bridge, Router, VLAN
- Real Time Scheduler für QoS
- Vollständige und Para-Virtualisierung

## Windows Beispiel Konfiguration:

```
kernel = "/usr/lib/xen/boot/hvmloader"  
builder='hvm'  
vcpu=1  
memory = 512  
name = "winxp"  
vif = [ 'type=ioemu, bridge=xenbr0', mac='a:b:c:d:e:f' ]  
disk = [ 'file:/export/winxp/hdc.raw,ioemu:hdc,w', 'phy:/dev/hda,hdc:cdrom,r' ]  
  
device_model = '/usr/lib/xen/bin/qemu-dm'  
boot="c"  
sdl=1  
vnc=0  
nographic=0
```

## Live Migration

- Für die Migration von VMs muss das VM Image von beiden Servern erreichbar sein.
- Bei Live Migration muss die Server Hardware identisch sein und das VM Image von beiden Server zugleich erreichbar sein.

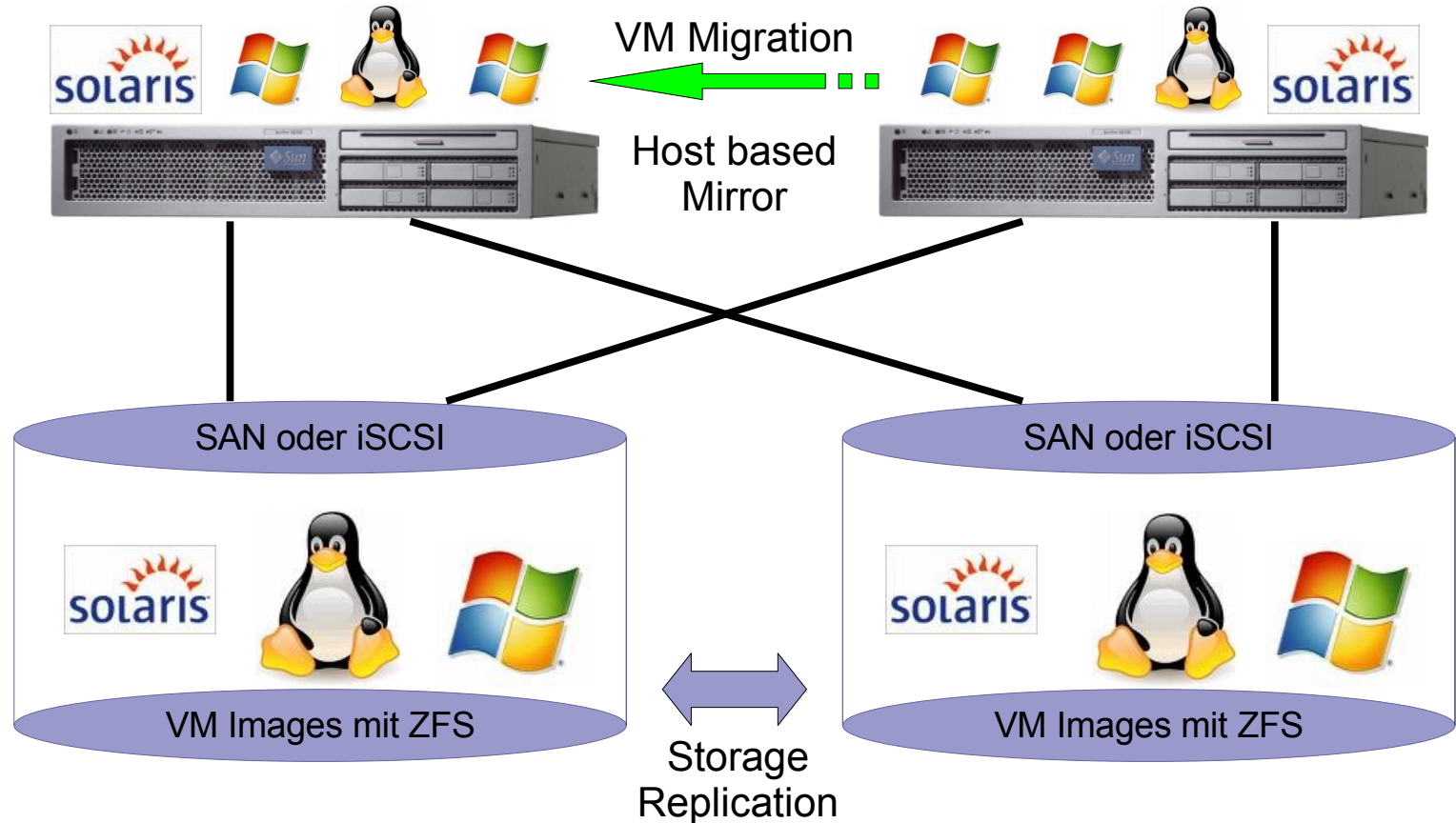


## Shared Storage:

- Raw Devices (LVM)
- Shared FileSystem
  - ◆ NFS
  - ◆ QFS
  - ◆ GFS

## HA DataCenter

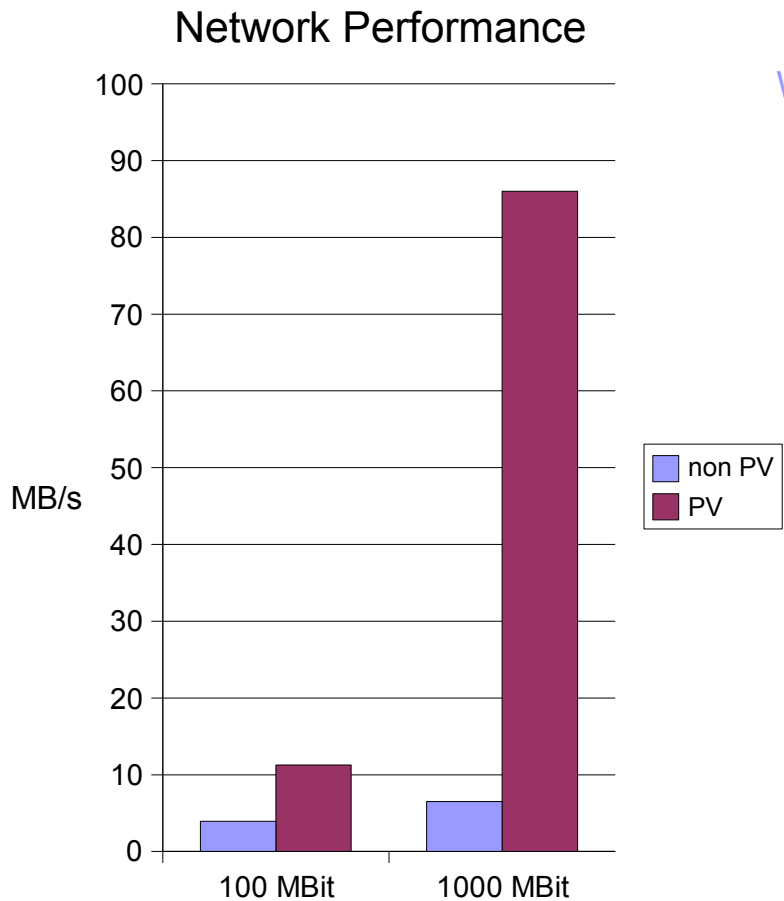
- VMs gespiegelt über 2 Storage Systeme und verteilt auf zwei Standorte
- Cloning und Backup von VMs mittels ZFS SnapShots
- „Kalt“ Migration für VMs, keine „Live Migration“ über 2 Storages



# Konfiguration

## Para-Virtuelle Treiber Performance

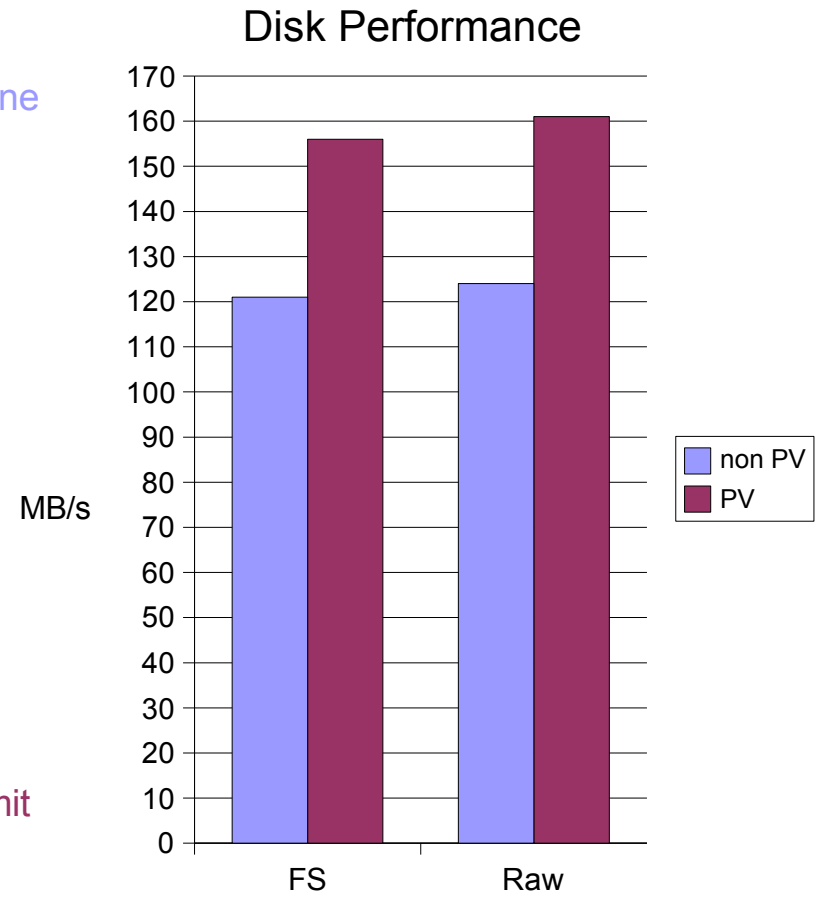
- Network Performance mit „netperf“ gemessen
- Disk Performance mit „iometer“ gemessen mit Raid0 16x140GB Disks (Sun 6140)



WindowsXP ohne PV Treiber

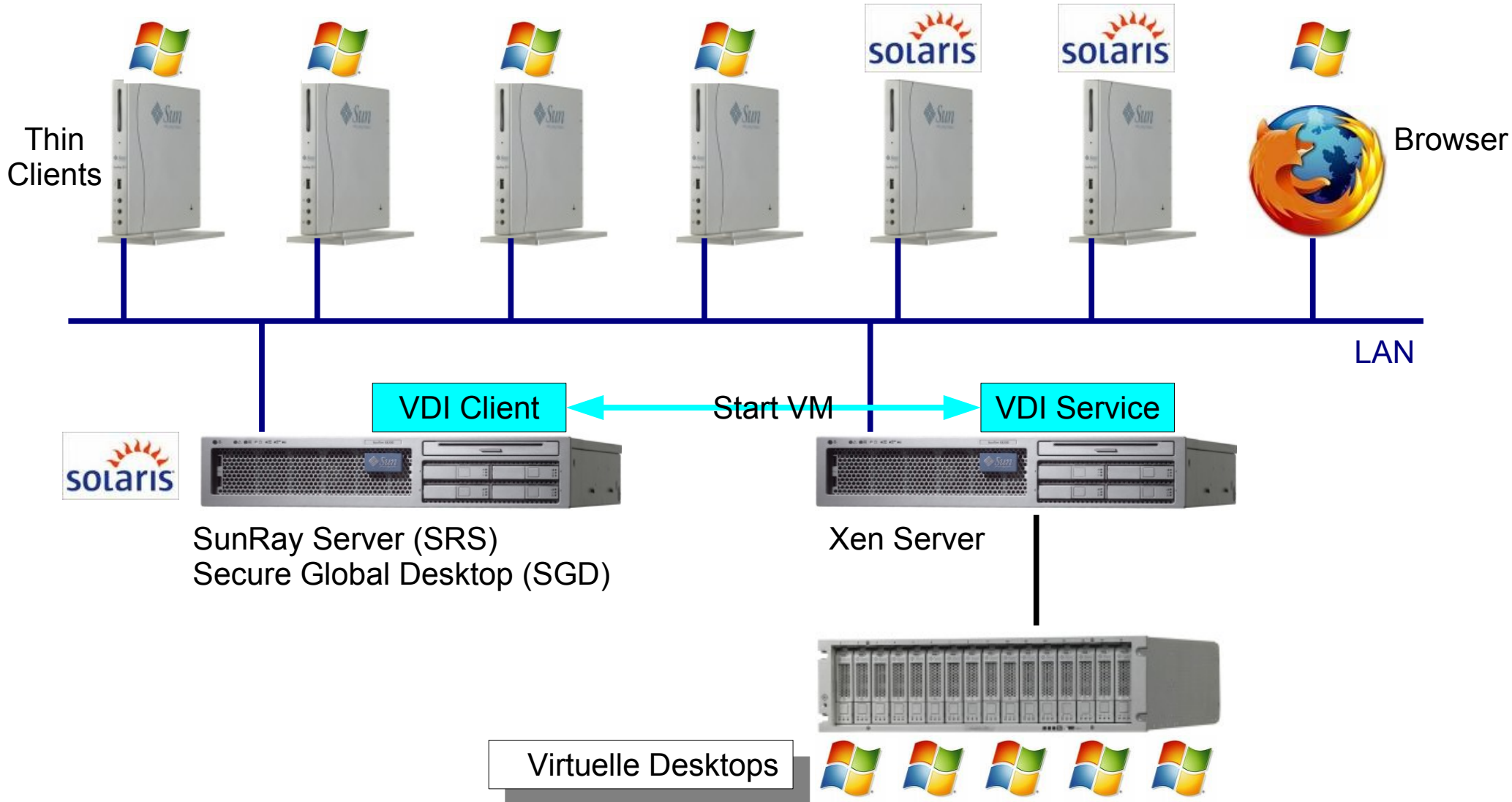


WindowsXP mit PV Treiber



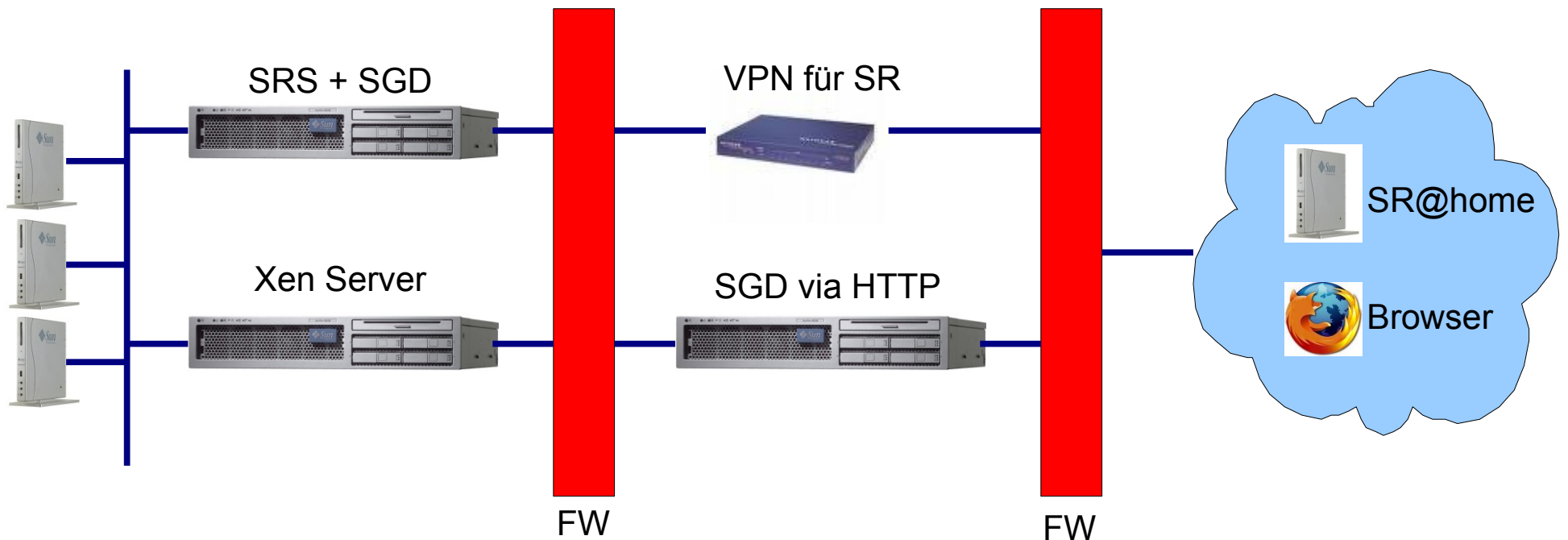
**Konfiguration**

**Virtuelle Desktop Infrastruktur**



### 3-Tier-Architektur

- Sicherer Zugriff auf virtuellen Desktop von überall
- Schnell arbeitsbereit durch Aktivieren der SmartCard
- Problemloser Arbeitsplatz Wechsel (Hot-Desking)



# Agenda

- Einführung und Übersicht
- Architektur
- Konfiguration
- Live Demo

**Danke !**



**Fragen ?**