Betriebssysteme und nebenläufige Programmierung

SoSe 2025

Roland Wismüller
Betriebssysteme / verteilte Systeme
roland.wismueller@uni-siegen.de
Tel.: 0271/740-4050, Büro: H-B 8404

Stand: 13. Mai 2025

Betriebssysteme und nebenläufige Programmierung

SoSe 2025

11 Virtualisierung

11.1 Virtualisierungstypen

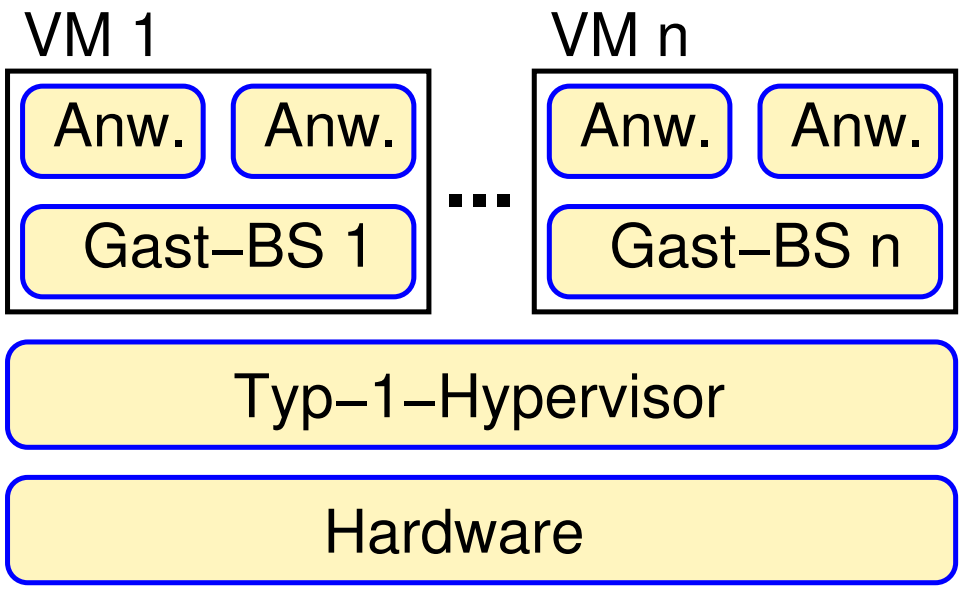
- Virtuelle Prozessoren: Emulator, z.B. QEMU, JVM
- Virtuelle Prozessumgebungen: normales BS
- Virtuelles BS: ABI für BS X auf BS Y, z.B. WINE
- Virtueller Desktop: z.B. RDP, VNC
- Virtuelle Ressourcen
  - z.B. *Storage Area Network* (SAN), virtuelle Netzwerke
- Virtuelles Laufzeitsystem (Sandboxing)
  - *Container*
- Virtuelle Computer (virtuelle Maschine)
  - wirkt (auch für BS) wie realer Computer

Motivation

- Server (WWW, Mail, ...) laufen i.a. auf eigenen Rechnern
  - sicherer und zuverlässiger
  - unterschiedliche Anforderungen an Systemsoftware / BS
- Statt einzelner Rechner: virtuelle Maschinen auf einem Rechner
  - kostengünstiger / energiesparender
  - Migration virtueller Maschinen (VMs) zwischen Rechnern möglich
  - Erneuerung der Hardware problemlos(er)
- Realisierung durch *Hypervisor* (*Virtual Machine Monitor*, VMM)
  - spielt BS auf VM vor, dass es direkt auf der Hardware läuft
  - VMs sind logisch getrennt; ggf. auf jeder VM anderes BS

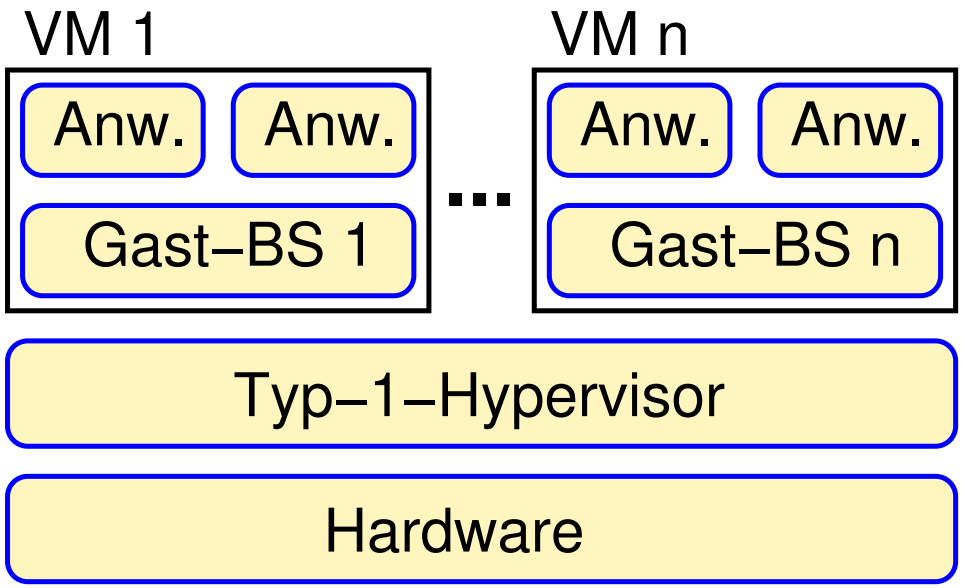
Schichtenmodell

Typ–1–Hypervisor

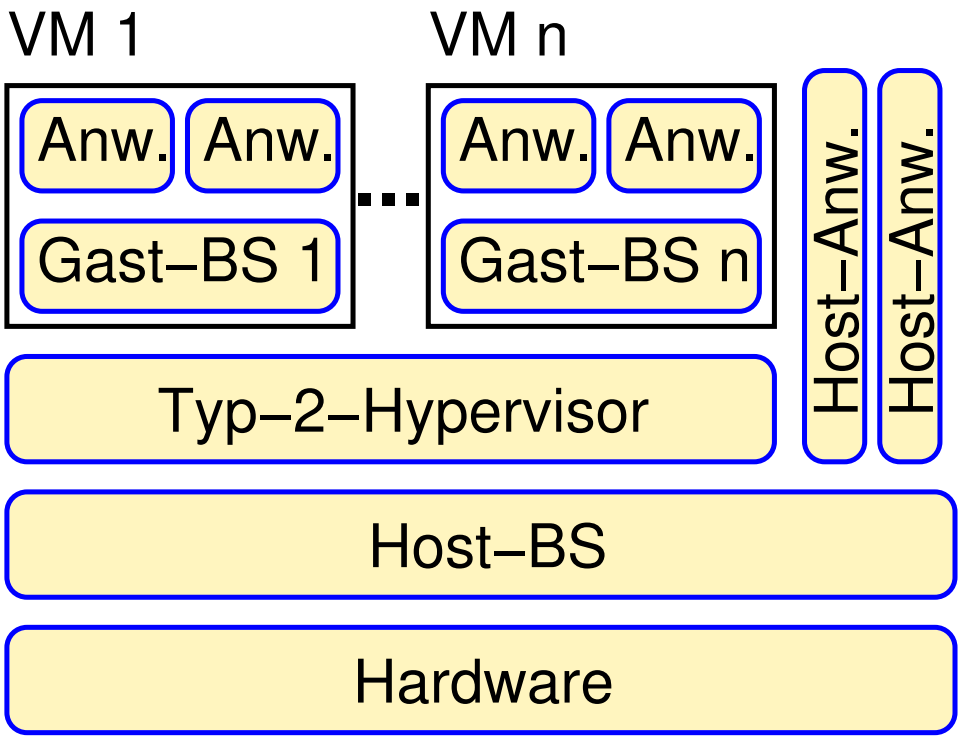


Schichtenmodell

Typ–1–Hypervisor



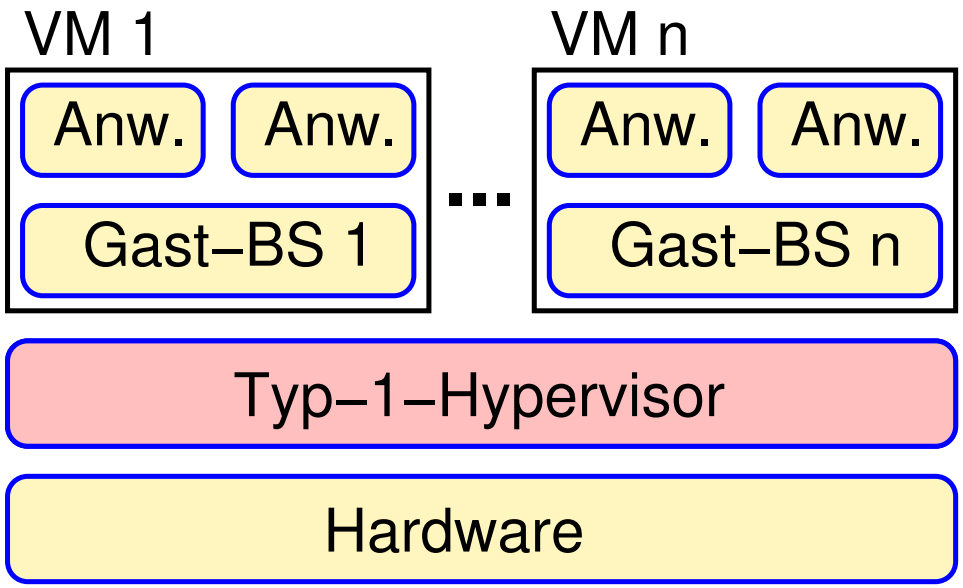
Typ–2–Hypervisor



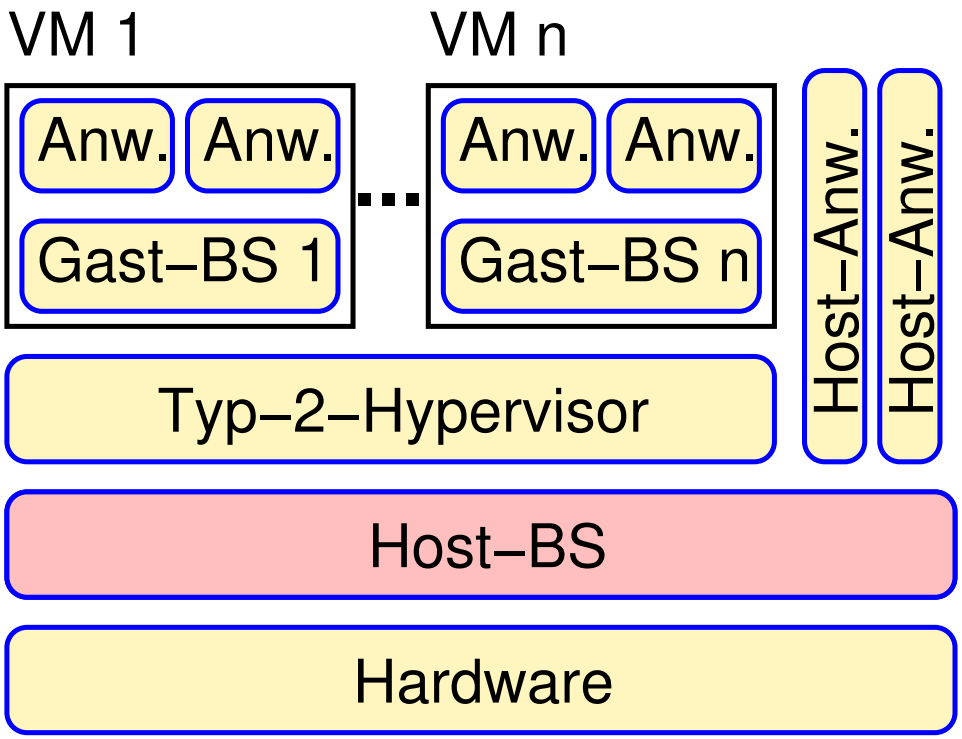
Mischform: Hypervisor im Host–BS integriert

Schichtenmodell

Typ–1–Hypervisor



Typ–2–Hypervisor

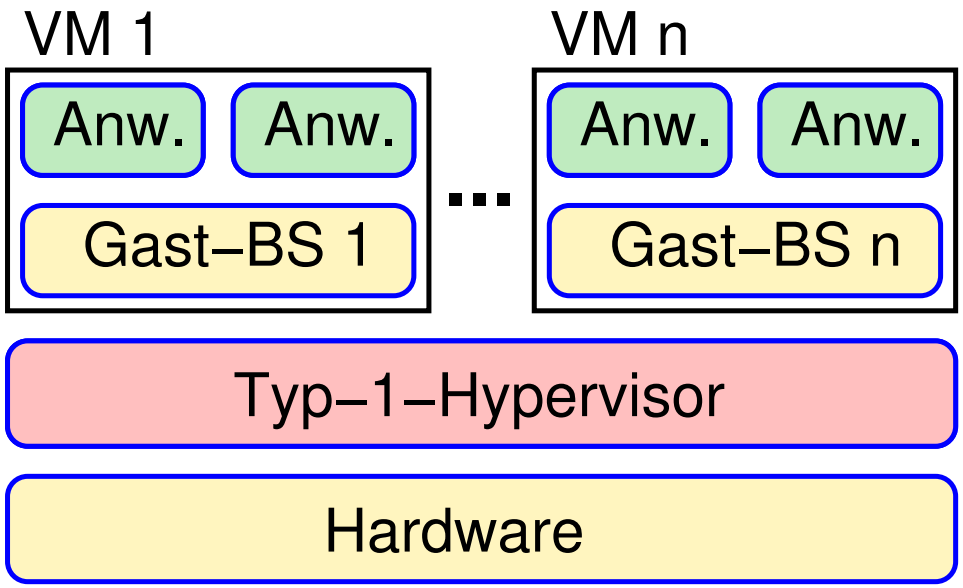


Systemmodus

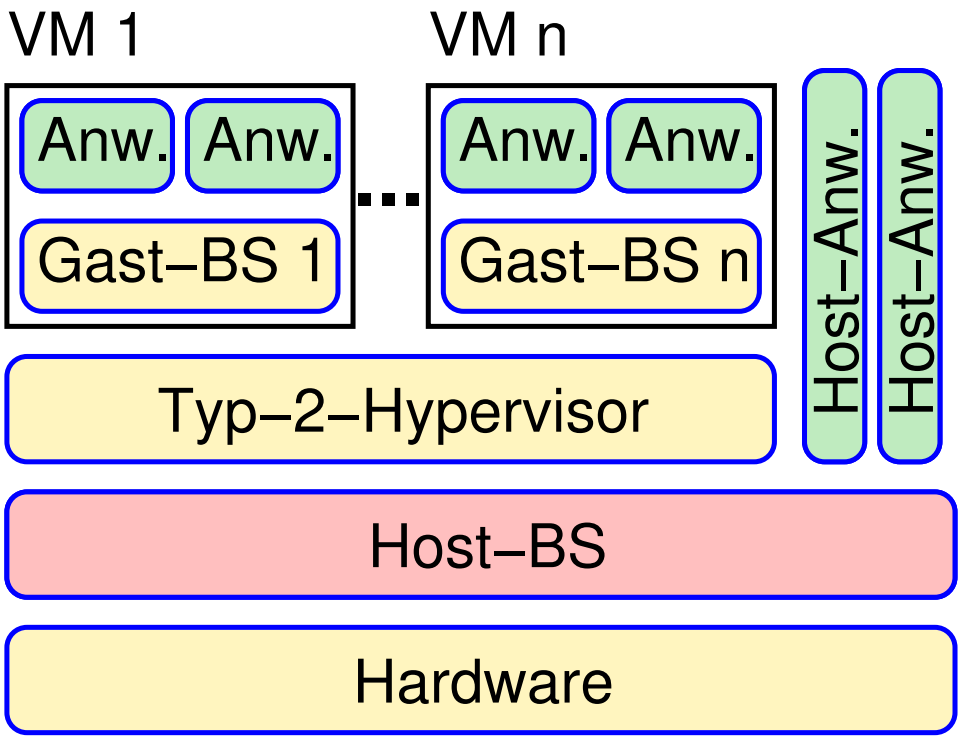
Mischform: Hypervisor im Host–BS integriert

Schichtenmodell

Typ–1–Hypervisor



Typ–2–Hypervisor

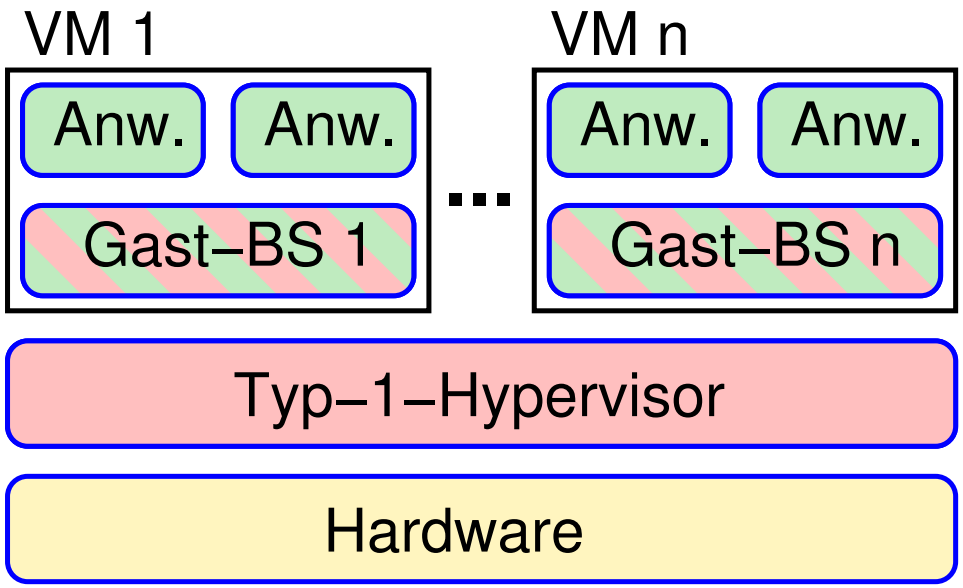


Systemmodus Benutzermodus

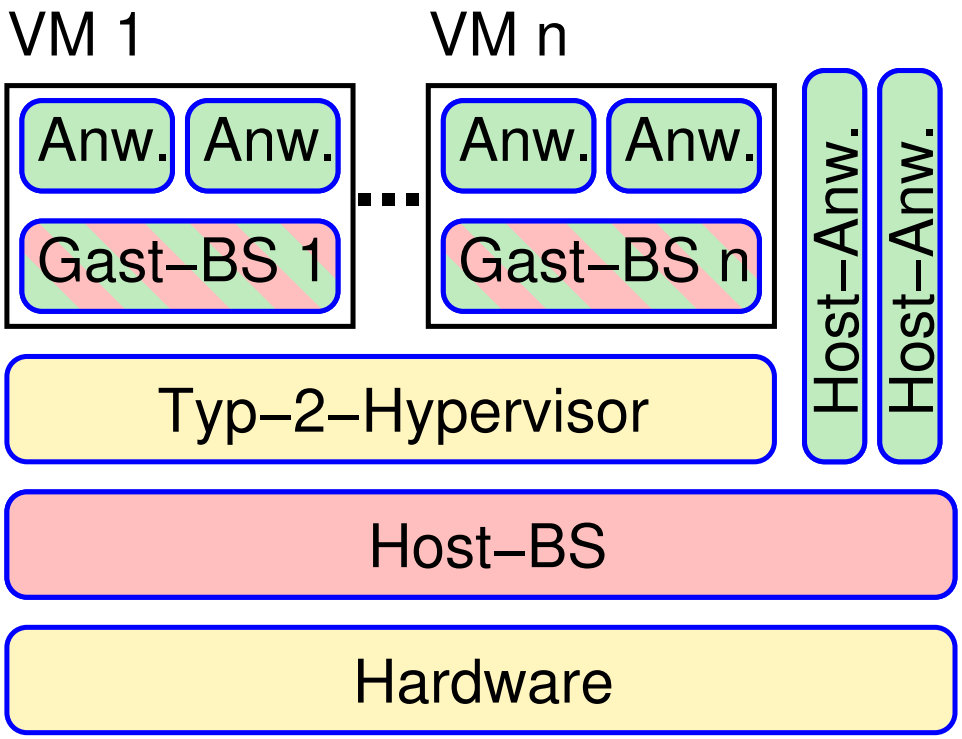
Mischform: Hypervisor im Host–BS integriert

Schichtenmodell

Typ–1–Hypervisor



Typ–2–Hypervisor

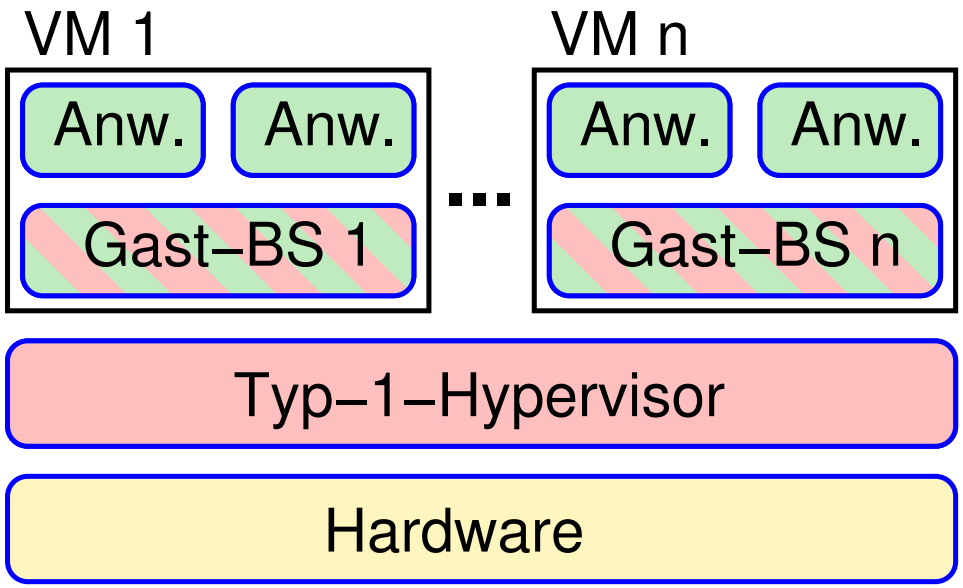


Systemmodus Benutzermodus Virtueller Systemmodus

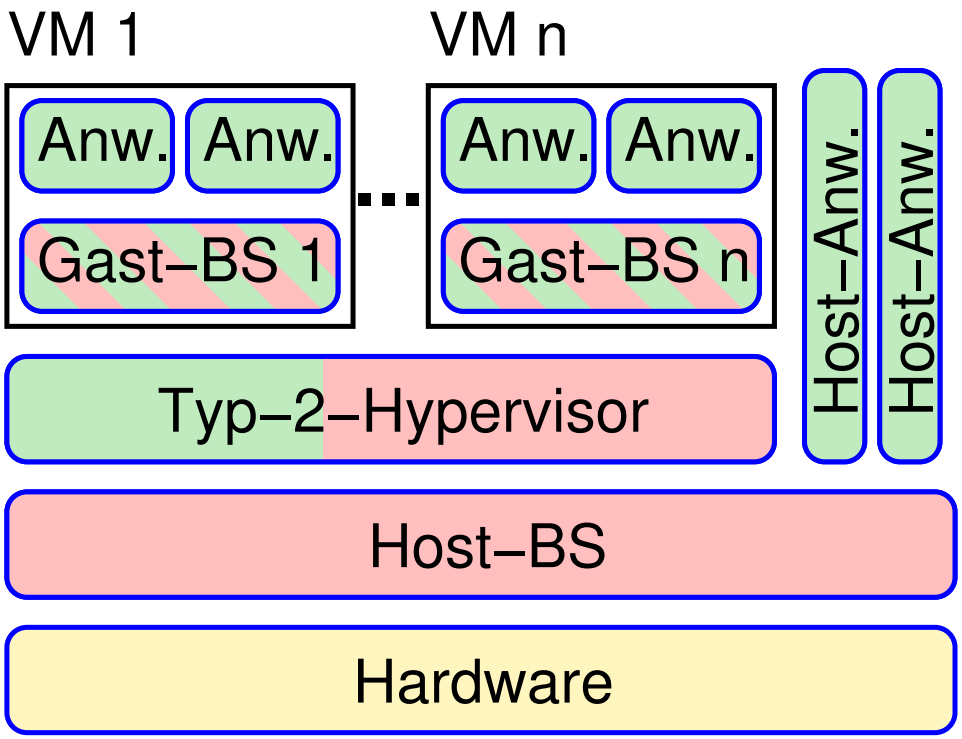
Mischform: Hypervisor im Host–BS integriert

Schichtenmodell

Typ–1–Hypervisor



Typ–2–Hypervisor



Systemmodus Benutzermodus Virtueller Systemmodus

Mischform: Hypervisor im Host–BS integriert

Beispiele

- Typ-I-Hypervisor
  - VMware ESX
- Typ-II-Hypervisor
  - VMware Workstation, VirtualBox, Xen
  - Xen kann durch eigenes Linux auch direkt auf HW aufsetzen
- In BS integriert:
  - KVM (Linux), Hyper-V (Windows)

Realisierung

- Gast-BS muss im Benutzermodus ausgeführt werden
  - Problem: Ausführung **sensitiver Befehle**
    - Verhalten im Benutzer- und Systemmodus unterschiedlich
- Sensitive Befehle müssen vom Hypervisor ausgeführt (emuliert) werden
- Einfach, falls alle sensitiven Befehle auch privilegiert sind
  - dann: Trap zum Hypervisor, dort Emulation
- Bei vielen (älteren) Prozessoren nicht der Fall (z.B. Pentium)
  - Lösung: Binärübersetzung des Codes
  - Alternative: **Paravirtualisierung**
    - Gast-BS wird angepasst

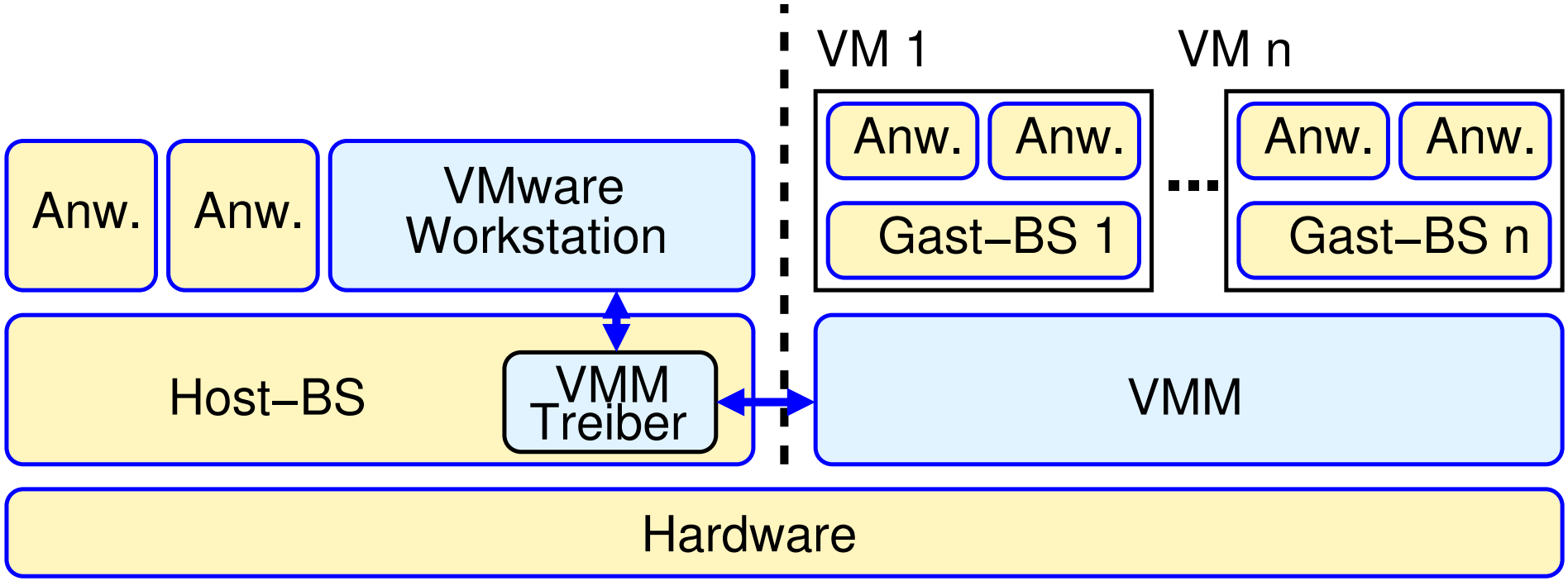
Binärübersetzung

- Idee: Code, der im Gastsystem im Systemmodus laufen muß, wird ersetzt
  - der neue Code beinhaltet keine sensitiven Befehle mehr
- Ersetzung erfolgt bei Bedarf auf Ebene von Basisblöcken
  - Basisblock: lineare Befehlsfolge, höchstens ein Sprungziel am Anfang, höchstens ein (i.a. bedingter) Sprung am Ende
  - Hypervisor hält einen Cache mit bereits übersetzten Basisblöcken
  - am Ende eines Blocks: Rückkehr zum Hypervisor
- Wird aus *Performance*-Gründen z.T. auch bei virtualisierbaren Prozessoren eingesetzt

Typ-2-Hypervisor

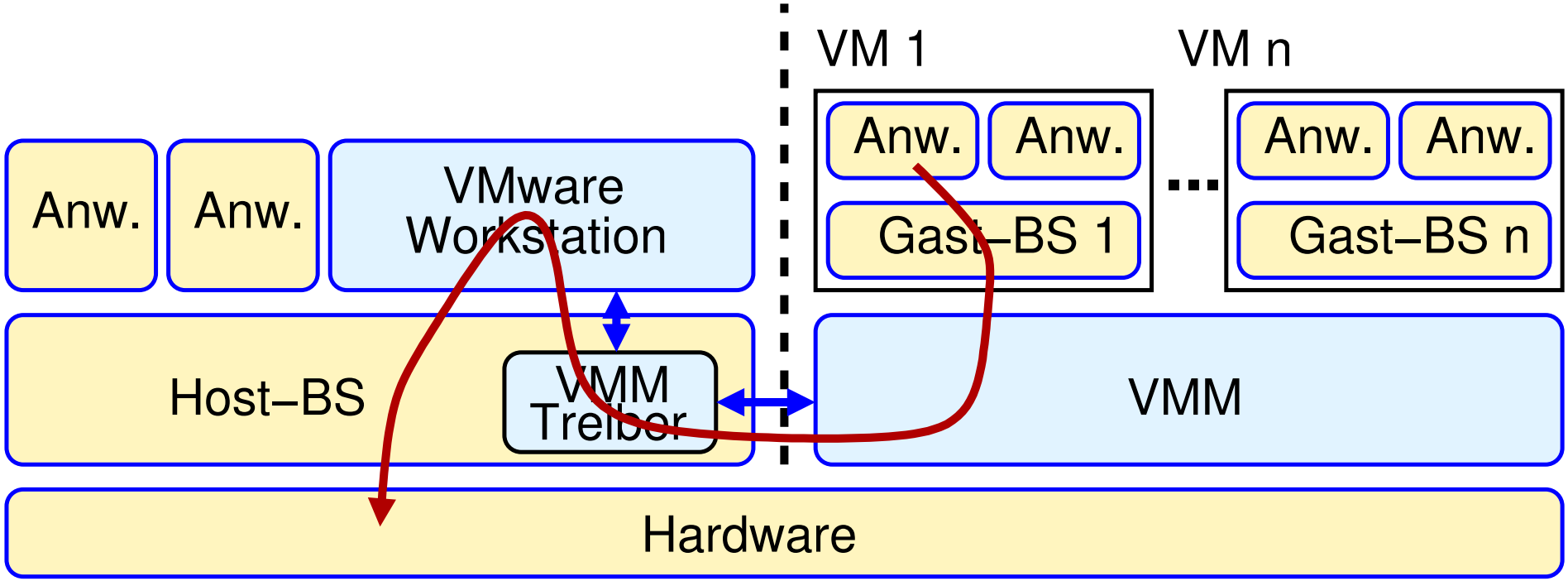
- Typ-2-Hypervisor ist „normale“ Anwendung
  - kann so aber CPU und Speicher nicht multiplexen
  - Hypervisor muss teilweise im Systemmodus laufen
- Daher: Aufspaltung in Anwendung, Gerätetreiber und VMM
- Anwendung: Bedienoberfläche, Realisierung von E/A über Host-BS
- Gerätetreiber: ermöglicht Ausführung des VMM im Systemmodus, emuliert Interrupts
  - *World-Switch*: Umschaltung zwischen Host-BS und VMM
- VMM: Multiplexing von CPU und Speicher, Binärübersetzung, Unterbrechungsbehandlung

Typ-2-Hypervisor: Beispiel VMware



- Ausgeführt wird entweder Host-BS oder VMM
  - *World-Switch* durch VMM-Treiber sichert Zustand des Host-BS

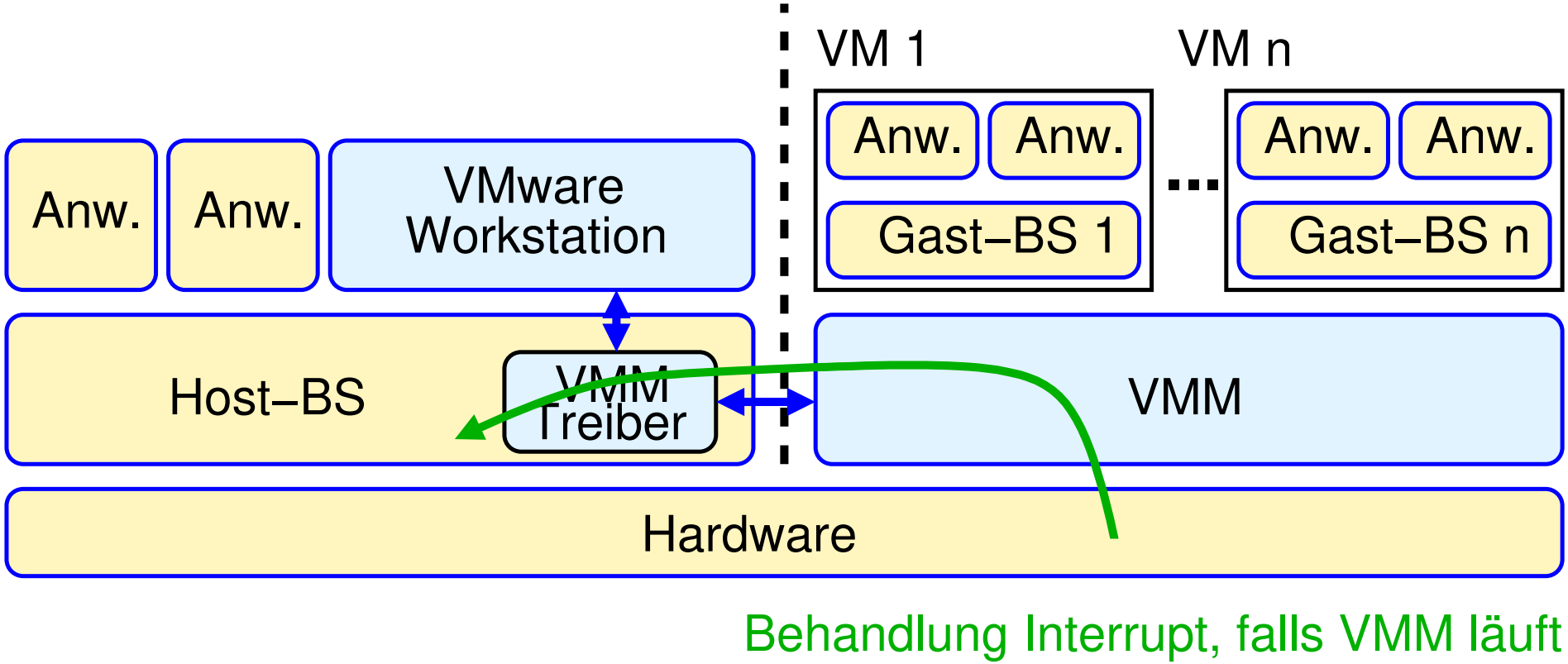
Typ-2-Hypervisor: Beispiel VMware



Routing einer E/A–Anfrage

- Ausgeführt wird entweder Host-BS oder VMM
  - *World-Switch* durch VMM-Treiber sichert Zustand des Host-BS

Typ-2-Hypervisor: Beispiel VMware



- Ausgeführt wird entweder Host-BS oder VMM
  - *World-Switch* durch VMM-Treiber sichert Zustand des Host-BS

Speichervirtualisierung

- Problem: jedes Gast-BS erstellt Seitentabellen unabhängig
  - dabei ggf. Abbildung auf dieselben Kacheln
- Hypervisor muß die Kacheln ggf. ändern
  - dazu: Schattentabellen im Hypervisor
- Problem: Hypervisor bekommt nur das Laden der Seitentabelle in die MMU mit, nicht aber spätere Veränderungen
- Mögliche Lösungen:
  - Seiten der Seitentabelle schreibschützen
  - Hinzufügen neuer Kacheln erst bei Seitenfehler
    - Löschen von Seiten mit privilegiertem Befehl (wegen TLB)
  - Hardwareunterstützung für verschachtelte Seitentabellen

Speichervirtualisierung ...

- Häufig: *Overcommitment*
  - alle VMs zusammen haben mehr Speicher als der Host
- Problem: Auslagerung durch Hypervisor nicht effizient / sinnvoll
  - unklar, welche Seiten für Gast-BS wichtig sind
  - Bei Auslagerung durch Gast-BS muss Hypervisor Seite ggf. erst wieder einlagern
- Lösung: *Balloon*-Treiber
  - im Gast-BS, kann ggf. fixierte Seiten allokieren
    - (fixierte Seiten können nicht ausgelagert werden)
  - Gast-BS muss dafür „normale“ Seiten auslagern
- Z.T. auch Reduktion des Speicherverbrauchs durch Deduplikation

E/A-Virtualisierung

- Gast-BS bekommt virtuelle (Standard-)Geräte
  - z.B. virtuelle SATA-Platte, realisiert als normale Datei
  - für Netzwerk realisiert Hypervisor ggf. auch einen virtuellen Switch
- Heute auch Geräte mit Hardware-Unterstützung für Virtualisierung
  - Single-Root-I/O-Virtualisierung (SR-IOV)
  - Gerätecontroller bieten jeder VM eine eigene Schnittstelle (virtuelle Funktionen)
    - keine Mitwirkung des Hypervisors nötig