1 Cloud Computing

- Cloud-Anbieter stellen skalierbar Hardwareressourcen zur Verfügung
- Software as a Service (SaaS) Anbieter stellen (darauf aufbauend) Dienste für Nutzer bereit

Eigenschaften von Cloud Computing Angeboten:

- Verfügbarkeit von scheinbar unbegrenzten Ressourcen
- Keine Kapazitätsplanung aus Sicht des Nutzers mehr nötig
- Pay-as-you-go Modell
  ↪ Kosten werden durch die tatsächliche Nutzung bestimmt

Resultat:

- economic of scale
  ↪ Geringere Anschaffungs- und Betriebskosten
  ↪ Höhere Auslastung als konventionelle Datenzentren
- Nutzer sparen Kosten und reduzieren finanzielle Risiken
  ↪ keine/wenig Hardware Anschaffung
  ↪ Bedarf wird nicht unter- oder überschätzt
  ↪ Neue Angebote können schneller auf den Markt gebracht werden

1.1 Basistechnologien

Virtualisierung

- Kann auf verschiedenen Ebenen erfolgen (z.B. Betriebssysteminstanz)
- Bessere Ausnutzung von Hardwareressourcen
  ↪ Ermöglicht mehrere Benutzer auf einer physischen Maschine
  ↪ Entkoppelung von Ausführungsort und Hardware
- Isolation von Nutzern

Web-Services

- Sprachunabhängige Basis für entfernte Kommunikation/Interaktion

1.2 Cloud Computing - Erweiterte Architektur

Software as a Service (SaaS)

- Vom Endnutzer oder anderen SaaS Einheiten genutzte Dienste

Function as a Service (FaaS)

- Ausführungsumgebung für einzelne Funktionen

Platform as a Service (PaaS)

- Konfektionierte Middleware für skalierbare Anwendungen

Infrastructure as a Service (IaaS)

- Mittels Virtualisierung bereitgestellte Ressourcen

1.3 Einsatzszenarien

Public Cloud

- Mieten von entfernten Ressourcen

Private Cloud

- Durch Virtualisierung flexibilisierte Verwaltung von Ressourcen
- Oder durch zusätzliche Mechanismen geschaffene Infrastruktur für einen Nutzer (z.B. Innerhalb von Amazon EC2)

Hyprides Cloud Computing

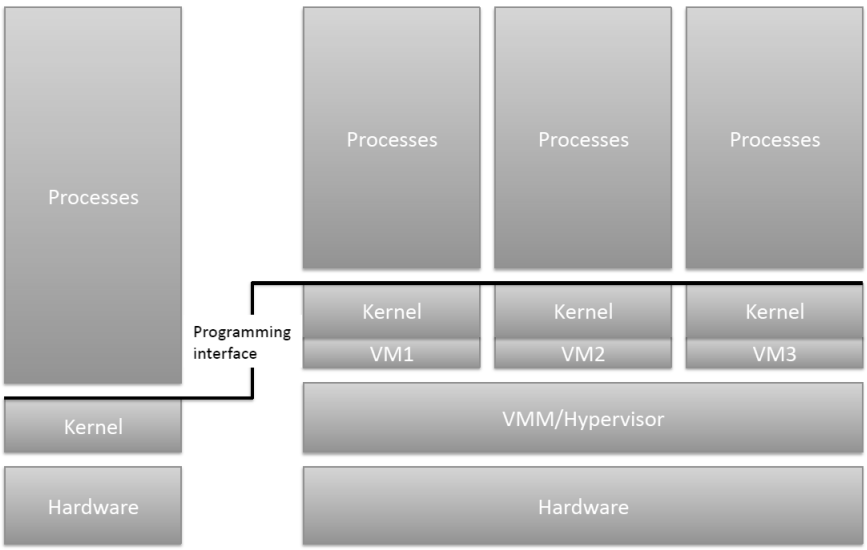
- Eigene Ressourcen mit denen einer öffentlichen Cloud kombinieren
- IT Infrastruktur schon vorhanden
- Kritische Daten, die nicht ausgelagert werden können
- Öffentliche Cloud nur zur Deckung von Bedarfsspitzen

Multi-cloud Computing

- Parallele Verwendung verschiedener Anbieter

2 Virtualisierung

Die Infrastruktur zur Virtualisierung wird **Virtual Machine Monitor (VMM)** oder **Hypervisor** genannt, die virtuelle Plattform entsprechend **Virtual Machine (VM)**.



Für eine virtuelle Maschine gelten folgende Bedingungen:

- **Äquivalenz**
  - ↪ Ein Programm verhält sich äquivalent zur direkten Ausführung
  - ↪ Tolerierte Abweichungen sind geringere Verfügbarkeit an Ressourcen und geändertes zeitliches Verhalten
- **Isolation**
  - ↪ VMs, die zusammen auf einer realen Plattform laufen, sollen effektiv voneinander isoliert sein
  - ↪ VMM hat die komplette Kontrolle über die Hardwareressourcen
  - ↪ VMs können nicht auf Ressourcen zugreifen, die ihnen nicht zugewiesen wurden
  - ↪ VMM kann unter bestimmten Umständen die Kontrolle über zugewiesene Ressourcen wieder entziehen
- **Effizienz**
  - ↪ Der weitaus größere Teil an Instruktionen des virtuellen Prozessors muss

direkt ohne Interaktion des VMMs, auf dem realen Prozessor ausgeführt werden

Letzte Anforderung ist nicht zwingen notwendig.

2.1 Ziele von Virtualisierung

- Bessere Ausnutzung existierender Ressourcen
- Erhöhung von Verlässlichkeit und Sicherheit
- Höhere Skalierbarkeit von Systemen
- Zentralisierung von Altsystemen ohne alte Hardware

2.2 Kritische Punkte bei der Virtualisierung

- Ausführung durch die CPU
  ↪ VMM emuliert eine CPU bzw. stellt eine virtuelle CPU zur Verfügung
- Zugriff auf Speicher mittels der MMU
  ↪ VMM muss den virtuellen Speicher emulieren (oder durch HW)
- Beim Zugriff auf Geräte werden Interrupts verwendet
  ↪ VMM muss Interrupts an Geräte weiterleiten
- Reaktion von Endgeräten (I/O) wird in Form von Interrupts vermittelt
  ↪ VMM muss die Interrupts in das System weiterleiten
- Zugriff auf Betriebssystem-Elemente unter Verwendung von Traps
  ↪ VMM muss Software-Interrupts an die entsprechende Stelle weiterleiten und die Verwaltung von Interrupt Tabellen virtuell gestalten
- Exceptions müssen behandelt werden können

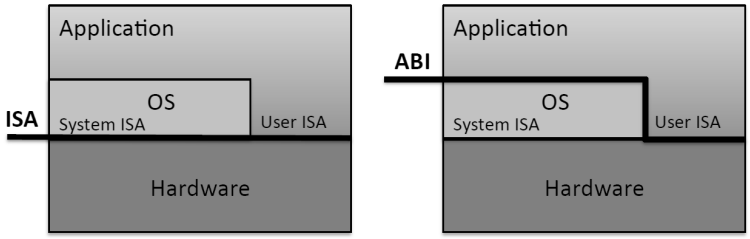
2.3 Ebenen der Virtualisierung

Instruction Set Architecture (ISA) bzw. Interface

- Schnittstelle der Hardware zum Betriebssystem und zu den Anwendungen
- System-virtuelle Maschinen virtualisieren das ISA

Application Binary Interfce (ABI)

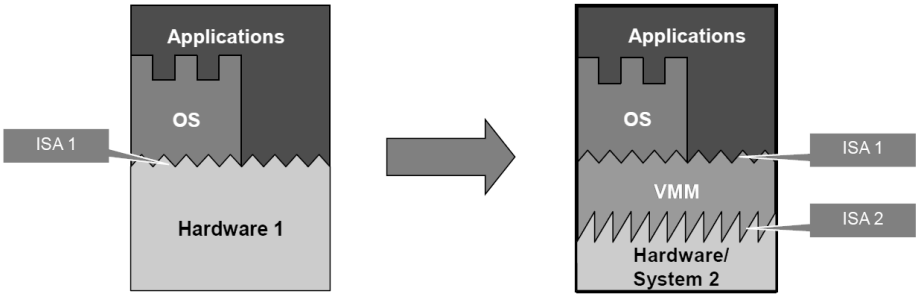
- Schnittstelle zum Betriebssystem, die Applikationen bzw. Prozesse nutzen
- Prozess-virtuelle Maschinen virtualisieren das ABI



2.4 System Virtualisierung

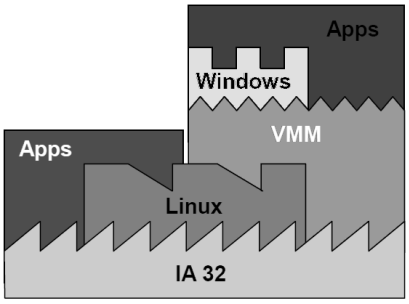
Vollständige Virtualisierung

- Keine Anpassung des Gastsystems nötig
- Unterstützung von fremder HW
  ↪ emuliert ISA auf fremder HW
- Beispiele: VMware Workstation, QEMU, Bochs



Hybrid Betrieb

– Beispiel: Linux läuft nativ und Windows wird in einer virtuellen Maschine ausgeführt



Hardware-basierte Virtualisierung

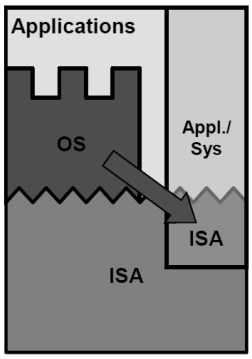
– Wenn die gleiche ISA virtualisiert wird kann zur Optimierung die ISA direkt verwendet werden

  ↪ Beispiele: VMware (Linux/Windows auf x86)

– Voraussetzung: Hardware-Unterstützung von Virtualisierung bzw. Schutz gewisser Einheiten durch VM

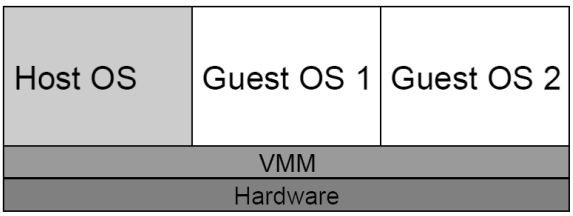
  ↪ Beispiele: Intel VMM/VMX

  – VMX-Instruktionen zum Generieren und Kontrollieren einer virtuellen Umgebung
  – VMM kontrolliert Interrupts und Exceptions
  – Memory Virtualisierung



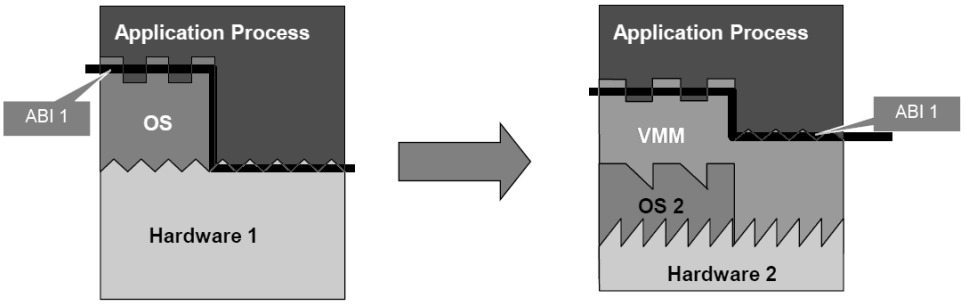
Paravirtualisierung

- Trennung von Host- und Gastbetriebssystem
- Gastbetriebssystem muss definierte Schnittstellen implementieren
  ↪ Erleichtert Kontrolle des Gastsystems
  ↪ Direkter Zugriff auf Hardware kann verhindert werden



2.5 Virtualisierung auf Ebene der ABI

- Stellt unifizierte Laufzeitumgebung unter evtl. verschiedenen Betriebssystemen zur Verfügung
- Beispiele: Betriebssystem-Virtualisierung (z.B. Docker), WINE und High Level Language Virtual Machines

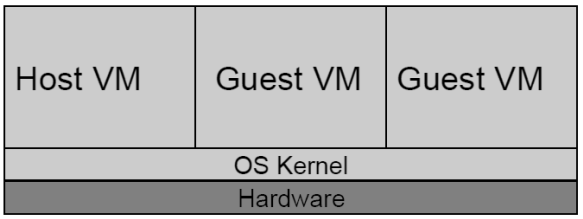


Betriebssystem-Virtualisierung

- Oft werden mehrere Betriebssystem-Instanzen inklusive des gleichen Kernels verwendet um verschiedene Dienste bereitzustellen
- Um Ressourcen zu sparen und hohe Effizienz zu erlangen ermöglicht Betriebssystem-Virtualisierung die gemeinsame Nutzung eines Kernels und dennoch eine isolierte Ausführung
- Verwendung des gleichen Kernels erfordert nur sehr geringe Kontextanpassungen und kann entsprechend hohe Performanz bieten
- Beispiel; VServer - aktuell: Docker

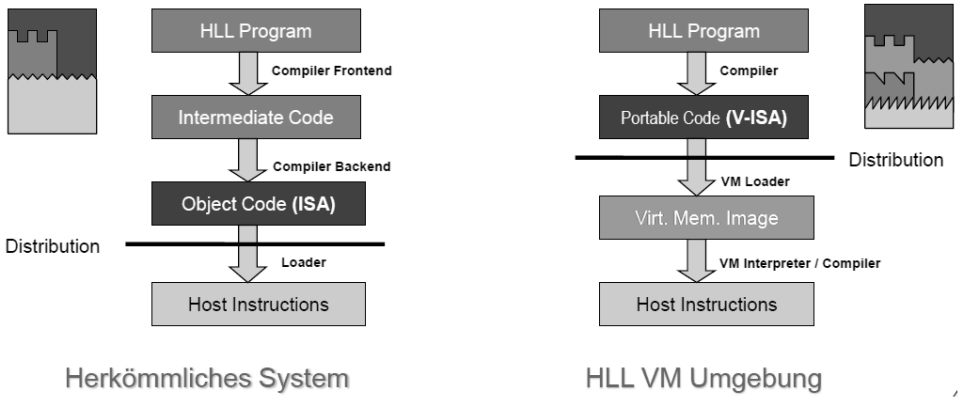
Anforderungen:

- Einsatz von Mechanismen zur Umsetzung von Ressource- und Sicherheitscontainern für Isolation



High Level Language (HLL) VMs

- Virtualisiert (in der Regel) keine reale Maschine
- Entwurf gemeinsam mit der Applikationsumgebung für die HLL
- Vorteile: Portabilität, zugeschnitten auf HLL
- Wird aktuell relevanter im Kontext Cloud Computing durch FaaS
- Beispiele: Java VM, JavaScript, .NET CLI



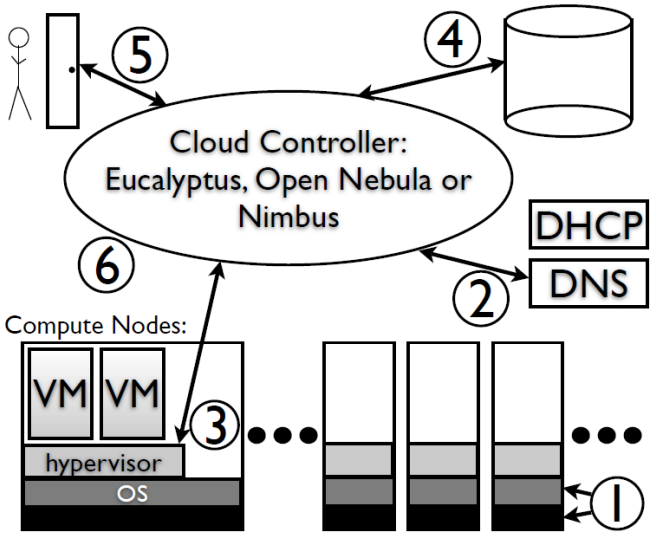
2.6 Aufbaue einer virtuellen Maschine

- Notwendige Betriebsmittel
  - Physische Maschine und Gastgeberbetriebssystem (Host)
  - Virtualisierungssoftware, die den VMM bereitstellt
  - Abbild der zu betreibenden virtuellen Maschine
- Ausbau des Abbilds einer virtuellen Maschine
  - Meta-Informationen (spezifisch, je nach Virtualisierungssoftware)
  - Dateisystem
    - Kern des zu virtualisierenden Gastbetriebssystem (Guest)
    - User-Space-Komponenten des Gastbetriebssystems
    - Daten

Analogie Objektorientierung:

- Klasse: statisches Abbild einer VM
- Instanz: eine im Betrieb befindliche VM

3 Infrastructure as a Service



1. Hardware und Betriebssystem
2. Netzwerk und Netzwerkdienste (z.B. DHCP, DNS)
3. Virtualisierung
4. Datenspeicher und Image-Verwaltung
5. Managementschnittstelle für Administratoren und Benutzer
6. Cloud-Controller

   ↪ Management der Ressourcen

Netzwerk und Netzwerkdienste

- Verwaltung und Einbindung der realen und der virtuellen Infrastruktur
- Zuordnung einer eindeutigen (virtuellen) MAC pro virtueller Maschine
- Vermittlung des Datenverkehrs zwischen der realen Maschine und den virtuellen Maschinen (bridge)
- Automatische Einbindung mittels DHCP (Zuordnung einer IP-Adresse) und DNS (Zuordnung eines Host-Namens)

– Eventuell Etablierung privater Netzwerke etc.

Umsetzung:

– Cloud-Controller muss bei Erzeugung einer virtuellen Maschine die Virtualisierungsinfrastruktur konfigurieren und entsprechende Informationen an den DHCP- sowie DNS-Server weitergeben

Virtualisierung

– Xen, KVM, VServer, etc.

– Im Open-Source-Bereich wird oft von der verwendeten Virtualisierungslösung mittels libvirt abstrahiert

Umsetzung libvirt:

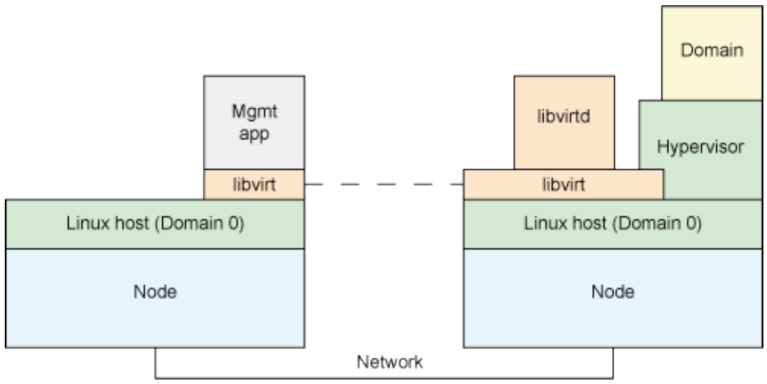
– Stellt einheitliche Schnittstelle zum Verwalten von unterschiedlichen Virtualisierungslösungen zur Verfügung

– Einfache C-Schnittstelle und Kommandozeilenunterstützung

  ↪ z.B. Starten, Stoppen, Migrieren und Pausieren von VMs

– Kommandos werden über libvirtd entgegengenommen

  ↪ Ermöglicht lokales wie auch entferntes Management von VMs



Datenspeicher und Image-Verwaltung

– Schnelle Erzeugung und De-/Installation von VM-Images

  ↪ Images müssen konfiguriert und eventuell um spezifische Software ergänzt werden

Umsetzung:

- Repository für VM-Images

  ↪ In der Regel Vorlagen, die erst noch in ein lauffähiges System umgewandelt werden (z.B. durch hinzufügen einer Swap-Partition oder Anpassung der Partitionsgröße)

Managementschnittstelle für Benutzer

- In der Regel grafische Benutzeroberfläche für den Anwender
  - Anfordern neuer VMs
  - Benutzerspezifische Konfiguration

    ↪ z.B. Amazon EC2: Größenklassen für VMs oder Ort der Ausführung
  - Verwaltung von Credentials für den sicheren Zugriff auf VMs

Cloud-Controller

- Zentrale Komponente einer IaaS-Infrastruktur
- Umsetzung der Benutzeranforderungen für die Erzeugung von VM-Images und ihre Installation
- Um-/Verteilung von VMs

4 Datenmanagement in Clouds

- Allgemeiner Trend: Extreme Zunahme der zu verarbeitenden Daten
  ↪ Induziert durch Zunahme an computergestützter Datenverarbeitung und exponentieller Zunahme der Kapazität von Speichermedien bei gleichen Kosten
- Problematik: Effiziente und skalierbare Verarbeitung von Daten
- Lösungsansatz: Neue Konzepte bzw. Algorithmen für verteiltes Datenmanagement
- Programmiermodell: MapReduce/Hadoop, Dryad
- Verteilte, einfache Datenbank-ähnliche Systeme bzw. Key-Value-Stores: Dynamo, BigTable, PNUTS
- Verteilte Dateisysteme: GoogleFS, HadoopFS

5 Web-Service

- Zentraler Begriff der Dienstleistung
  ↪ Komponente bietet Dienst an
- Zugang zu Web-Services über das Web
  ↪ Nutzung standardisierter Protokolle und Konzepte
- Vision vom Markt der Komponenten (Web-Services)
  ↪ Unabhängige Softwarefirmen verkaufen Web-Services-Software
  ↪ Web-Services sind nicht ortsgebunden
    ↪ Kann aus der Entfernung zugegriffen werden
    ↪ Benötigt in der Regel keine Installation von Software für den Nutzer
- Web als einfache Schnittstelle

Definition Web-Services

Ein Dienst, dessen Schnittstelle mit **WSDL** beschrieben, mit **UDDI** registriert und gefunden und mit **SOAP** angesprochen werden kann.