Arithmetik

Was sind die Sonderformate bei IEEE

Einfache Genauigkeit		Doppelte Genauigkeit		Dargestelltes Objekt
Exponent	Bruch	Exponent	Bruch	--
0	0	0	0	
0	Nicht 0	0	Nicht 0	
1 bis 254	Beliebig	1 bis 2046	Beliebig	
255	0	2047	0	
255	Nicht 0	2047	Nicht 0	

Arithmetik

Einfache Genauigkeit		Doppelte Genauigkeit		Dargestelltes Objekt
Exponent	Bruch	Exponent	Bruch	
0	0	0	0	0
0	nicht null	0	nicht null	± denormalisierte Zahl
1–254	beliebig	1–2046	beliebig	± Gleitkommazahl
255	0	2047	0	±unendlich
255	nicht null	2047	nicht null	NaN (Not a Number)

Arithmetik

Was ist die Vorzeichenbetrags schreibweise?

Arithmetik

MSB 0 oder 1 gibt vorzeichen an

Arithmetik

Wie funktioniert die Zweierkomplementdarstellung?

Arithmetik

- Negative Zahlen werden durch Invertieren aller Bits und Addieren von 1 gebildet.

Cache

Welche auswirkung hat die änderung der Cachegröße?

Cache

Vergrößerung	Vorteile	Nachteile
↑ **Cache-Größe**	↓ Miss Rate (mehr Daten passen rein)	↑ Latenz (längere Zugriffszeit) ↑ Energieverbrauch & Flächenbedarf
↓ **Cache-Größe**	Schneller (kürzere Zugriffszeit) Geringerer Stromverbrauch	↑ Miss Rate (häufiger Hauptspeicherzugriff)

Cache

Welche auswirkung hat die änderung der Assozivität auf den Cache?

Cache

Assoziativität	Vorteile	Nachteile
↑ **Höhere Assoziativität** (z. B. 8-Wege)	↓ Conflict Misses Besser für unvorhersehbare Zugriffsmuster	↑ Latenz (mehr Vergleiche nötig) ↑ Energieverbrauch (komplexere Hardware)
↓ **Niedrigere Assoziativität** (z. B. Direct-Mapped)	Schneller (nur 1 Tag-Vergleich) Geringere Hardware-Kosten	↑ Conflict Misses (Kollisionen bei gleichem Index)

Cache

Wie berechnet sich der Offset?

Cache

Offset-Größe (Bits)=log2(Blockgröße in Bytes)

Cache

Wie berechnet sich die gesamtgröße des Cache?

Cache

Cache-Größe (Bytes) = Anzahl der Blöcke × Blockgröße (Bytes)
Anzahl der Blöcke = Anzahl der Sets ×
Assoziativität (Blöcke pro Set)

Cache

Wie funktioniert Write Through und wie kann man seine Probleme fixen?

Cache

Speicher immer direkt mit aktualiseren
Aber Schreibzugriffe deswegen immer sehr langsam
-> Write bufffer und erst schreiben wenn Write Buffer voll ist

Cache

Wie berechnet sich die Anzahl der Seitentabellen einträge bei Virtuellem Speicher?

Cache

Anzahl PTEs = Virtueller Adressraum /
Seitengröße =2^(Adressgröße−Offset-Bits)

Cache

Was ist der unterschied zwischen Cache Kohärenz und Konsistenz?

Cache

Kohärenz:
Wann muss ein Wert aktualisiert/invalidiert werden?
Konsistenz:
In welcher Reihenfolge müssen Schreib-/Lesezugriffe sichtbar sein?

Cache

Wie berechnet sich der Index?

Cache

$\log_2$(Anzahl der Cache-Blöcke)

Cache

Wie funktioniert Write-Back?

Cache

Speicher wird nur aktuallisiert wenn ein block mit dirty bit evicted wird. Dirty bit wird bei einem Cache write gesetzt. Benötigt verwaltungslogic aber ist schneller als write through.

Cache

Wie berechnet sich die Größe des virtuellen Adressraumes?

Cache

Virtueller Adressraum (Bytes)=2^Adressgröße(bits)

Cache

Wie berechnet sich die größe einer Seitentabelle?

Cache

Seitentabellen-Größe (Bytes)=Anzahl Einträge × Größe Einträge

Cache

Welche auswirkung hat die änderung der Blockgröße?

Cache

Vergrößerung	Vorteile	Nachteile
↑ **Blockgröße**	↑ Räumliche Lokalität (weniger Conflict Misses) Gut für sequentielle Zugriffe (z. B. Vektorverarbeitung)	↑ Latenz (mehr Daten transferieren) ↑ Cache-Verschwendung (wenn nur kleine Daten genutzt werden)
↓ **Blockgröße**	Geringere Transfer-Latenz Weniger Verschwendung bei kleinen Daten	↑ Miss Rate (mehr Capacity/Conflict Misses)

Cache

Was für Kohärenz Protokolle gibt es ud was machen diese?

Cache

Snooping Protokolle: Alle Caches "lauschen" auf Bus-Transaktionen.
Verzeichisbasiert: Zentrale Tabelle ("Directory") trackt Cache-Zustände.

Cache

Was sind Synchronisatiosnprotokolle bei Kohärenz und was machen diese?

Cache

Write Invalidate:

- Wenn ein Kern in eine gemeinsame Speicheradresse schreibt, werden **alle anderen Cache-Kopien invalidiert**.
- Andere Kerne müssen bei ihrem nächsten Lesezugriff die **aktualisierten Daten neu anfordern**.

Write Update/Broadcast:

- Bei einem Schreibzugriff wird der **neue Wert an alle anderen Caches gesendet**.
- Diese **aktualisieren ihre Kopien sofort**.

Formeln

Formel Leistung

Formeln

Durchsatz

$$Leistung = \frac{1}{Ausführungszeit}$$

X ist n mal schneller als z

$$\frac{LeistungX}{LeistungY} = \frac{AusführungszeitY}{AusführungszeitX} = n$$

Formeln

Amdahls Gesetz

FormeIn

$$T_{improved} = \frac{T_{affected}}{improvement\ factor} * T_{unaffected}$$

F = Anteil des Programmes

Formeln

Pipeline Speedup

Formeln

$$Zeit\ zwischen\ Befehlen_{mitPipeline} = \frac{Zeit\ zwischen\ Befehlen_{ohnePipeline}}{Anzahl\ der\ Stufen}$$

Formeln

Formel CPU-Zeit

Formeln

CPU-Zeit:

- Vergangene Zeit

$$CPU\ Zeit = Benutzer\ CPU\ Zeit + System\ CPU\ Zeit$$

$$CPU\ Zeit = Anzahl\ CPU\ Taktzyklen * Taktdauer$$

$$= \frac{Anzahl\ CPU\ Taktzyklen}{Taktfrequenz}$$

$$CPU\ Zeit = \frac{Befehle}{Programm} * \frac{Taktzyklen}{Befehl} * \frac{Sekunden}{Taktzyklus}$$

Formeln

Speedup Amdahls Gesetz

Formeln

$$Speedup = \frac{1}{\left(1 - F_{parallelisierbar}\right) + \frac{F_{parallelisierbar}}{100}}$$

F = Anteil des Programmes

Formeln

Formel CPI

Formeln

Cycles per Instruction

$$CPU\ Zeit = Befehlszähler * CPI * Taktdauer = \frac{Befehlszähler * CPI}{Taktfrequenz}$$

FormeIn

Formel Memory stall cycles

FormeIn

$$\text{Memory stall cycles}$$

$$= \frac{\text{Memory accesses}}{\text{Program}} \times \text{Miss rate} \times \text{Miss penalty}$$

$$= \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Misses}}{\text{Instruction}} \times \text{Miss penalty}$$

Formeln

AMAT

Formeln

AMAT = Hit Rate * Hit Time + Miss rate * Miss penalty

Formeln

Anzahl CPU Taktzyklen

Formeln

$$Anzahl\ CPU\ Taktzyklen = Befehlszähler * CPI$$

Formeln

MTBF

Formeln

MTBF = MTTF + MTTR

Formeln

Verfügbarkeit

Formeln

MTTF/(MTTF+MTTR)

Formeln

Anual Failure Rate

Formeln

$$\text{Annual Failure Rate (AFR)} = \frac{1000 \text{ disks} \times 8760 \text{ hrs/disk}}{1200000 \text{ hrs/failure}} = 0.73\%$$

Generel

Was versteht man unter Kontrollpfad

Generel

Kontrolliert den Datenpfad und gibt an, wann was gemacht wird.

Generel

Was besagt Moores law

Generel

- Anzahl der **Transistoren pro Chips** verdoppelt sich ca. alle 18 Monate
- -> Logaritmischer Wachstum
- Gilt immernoch (Nicht verwechseln mit Leistungswachstum)

Generel

Was versteht man unter der Powerwall?

Generel

- Mit höherer Taktfrequence wird die Wärmedichte immer höher
- -> Mehr Kühlung benötigt
- Blockiert die erhöhung der Taktfrequenz
- -> Lösung: Multiprozessoren

Generel

Was besagt Amdahls Gesetz?

Generel

- Manerwartet proportionale Verbesserung der Gesamtleistung bei Verbesserung
  eines Computeraspektes
  - Verbesserung wird nur dort spürbar wo sie angewendet wird
    → Make the common case fast
- Gesamtleistung ist immer von unveränderten Teil begrenzt

Generel

Was versteht man unter Datenpfad

Generel

Die Strecke über die in der Hardware benötigten Bausteine

Generel

Was versteht man unter kritischer Pfad

Generel

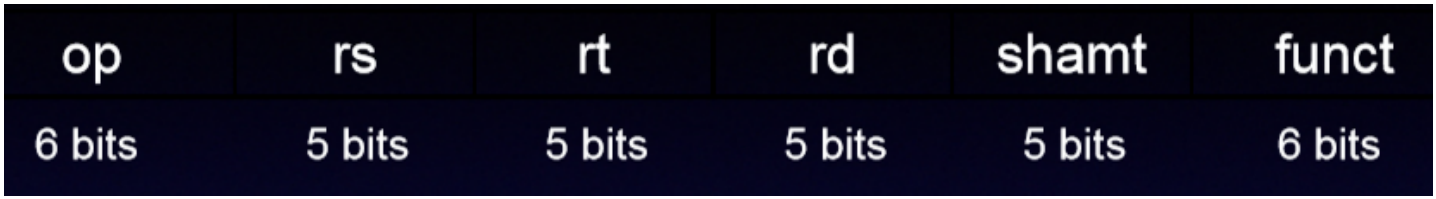
Der kritische Pfad ist die längste Abfolge an vorgängen. DIese bestimmt die maximale Taktfrequenz.

Mips

Wie ist das R-Format aufgebaut mit nutzen und erklärung

Mips

Für Register befehle



- Op = operation code (R-Format, S-Format…)
- Rs = first source Register
- Rt = second source Register
- Rd = Destiation register
- Shamt = shift amount (sll, srl)
- Funct = function code (add, sub…)
- Immediate = Zahlwert

Mips

Was sind die drei Adressierungsformate in der Mips?

Mips

R-Format
I-Format
Pseudo direkte Sprungadressierung

Mips

Wie ist das I-Format aufgebaut mit nutzen und erklärung?

Mips

- Für Immediate und branch befehle

op	rs	rt	Immediate
6 bits	5 bits	5 bits	16 bits

- Für PC relative Jumps (PC = PC + Immediate * 4)

Mips

Wie ist die Pseudo direkte Sprungadressierung aufgebaut mit nutzen und erklärung?

Mips

- Springe zu einer bestimmten addresse
- Wird für J und Jal benutzt

- PC = address * 4

Mips

Was sind die vier Entwurfsprinzipien einer ISA mit erklärung?

Mips

Einfachheit begünstigt Regelmäßigkeit

- Regelmäßigkeit macht Implementierung einfacher
- Einfachheit ermöglicht höhere CPU Leistung

Kleiner ist schneller

- Weniger suchzeit bei kleinen Speichern -> kleiner Register/Cache

Make common case fast

- Kleine Konstanten sind häufig
- Immediate Befehle erlauben load zu vermeiden

Good design demands good compromises

- Verschiedene Formate komplizieren Decodierung aber erlauben überall 32 bit Befehle

Multiprozessoren

Was ist mit starker und schwacher Skalierung gemeint und wodurch sind diese begrenzt (mit formel)?

Multiprozessoren

1. Starke Skalierung (Strong Scaling)

- *Wie schnell löst man ein **festes Problem**, wenn man mehr Prozessoren/Kerne hinzufügt?*
- **Grenzen**: Kommunikations-Overhead, Amdahls Gesetz.
- Speedup = T_1/T_N
- T_1 = Zeit auf 1 Kern, T_N = Zeit auf N Kernen

2. Schwache Skalierung (Weak Scaling)

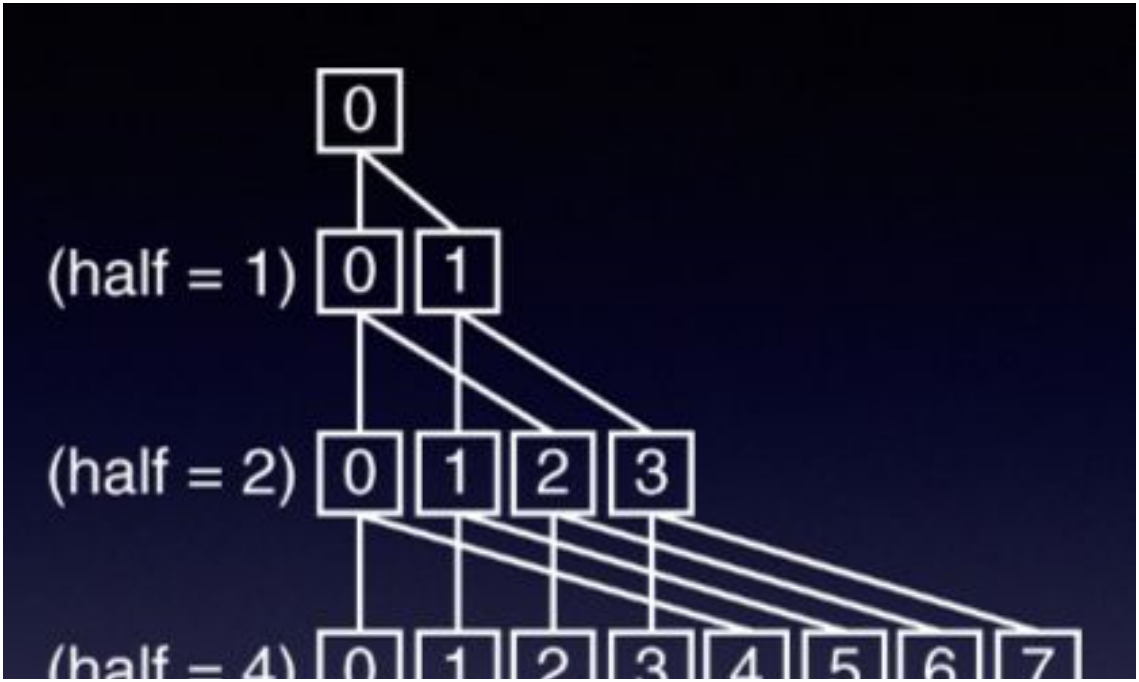
- *Wie viel größeres Problem kann man lösen, wenn man mehr Prozessoren/Kerne hinzufügt?*
- **Grenzen:** Speicherbandbreite, Lastverteilung.
- Effizienz = Problemgröße pro Kern / Gesamtzeit

Multiprozessoren

Was ist Summenreduktion und wie funnktioniert diese?

Multiprozessoren

Dafür benötigt man einen SMP (Shared Memory multiprocessor)
Hier werden z.B. immer zweierpaare an werten Addiert und danach wieder das selbe, bis nurnoch ein wert übrig ist



Pipeline

Welche formate benutzen welche Teile der Pipeline?

Pipeline

R-Format: IF ID(2 registerwerte lesen) EX(ALU) WB(Ergebnis schreiben)
I-Format: IF ID(registerwert lesen) EX(Adresse aus 16-bit Offset berechnen) MEM(load oder store) WB(nur bei load)
J-Format: IF ID(registerwerte lesen) EX(vergleich bei branch und zieladresse berechnen)

Pipeline

Was sind die Pipeline schritte in der Mips?

Pipeline

1. **IF**: Instruction Fetch vom Speicher
2. **ID**: Instruction Decode & Register lesen
3. **EX**: Execute Befehl ausführen oder Adresse ausrechnen
4. **MEM**: Zufriffe auf den Speicheroperanden
5. **WB**: Write back (Ergebnis in Register schreiben)

Pipeline

Was gibt es für Pipeline Konflikte und wie kann man diese lösen?

Pipeline

1. Strukturkonflikt (Eine benötigte Ressource ist belegt)
   - -> Datenpfade mit Pipelining erfordern getrennte Befehls- und Datenspeicher
2. Datenkonflikt (Müssen warte, das vorherige Befehle das Lesen/Schreiben von Daten abschliesst)
   - -> Forwarding, bei Load-Use mit stall oder code scheduling
3. Steuerkonflikt (Steuerentscheidung hängt vom vorherigen Befehl ab)
   - -> Delayed branch slot, branch prediction

Pipeline

Was ist der unterschied zwischen Statischer und dynamischer branch prediction?

Pipeline

- **Statische Sprungvorhersage:**
  - Basierend auf typischem Sprungverhalten (Festgelegte Regeln)
- **Dynamische Sprungvorhersage:**
  - Hardware misst tatsächliches Sprungverhalten (z.B. durch speichern der letzten Sprünge für jeden Branch)
  - -> Annahme: Zukünftiges Verhalten setzt Trend fort
  - Wenn falsch, stall während erneuten Befehlsladens und Aktualisierung der Branch history table

Pipeline

Wozu dient die in Prozessoren Sprungvorhersage und warum wurde sie entwickelt? Erklären Sie die zwei grundlegenden Alternativen Sprungvorhersage zu realisieren. Erklären Sie im Detail was in beiden Alternativen bei einem Sprungbefehl passiert und welche Prozessorressourcen dafür benötigt werden.

Pipeline

Die Sprungvorhersage trifft eine Vorhersage ob ein Sprung genommen wird oder nicht genommen wird.

- (1) Sie dient dazu, Pipelineverzögerungen zu vermeiden die entstehen, wenn der nächste Befehl nicht zum Dekodieren bereit steht – der Grund, weshalb sie entwickelt wurde.
- (2) Bei der statischen Vorhersage wird immer eine Richtung gewählt, z.b. für back brahöhes ein taken für forward branches ein not-taken. Es ist dazu keine weitere Ressource benötigt, da der Compiler (oder Assemblerprogrammierer) ein Vorhersagbit setzt.
- (2) Bei der dynamischen Vorhersage wird die Befehlsadresse als Index in eine Branch History Table verwendet. Der Zustand des 1 Bit oder 2 Bit oder komplizierteren Predictors ist dort gespeichert und wird zur Vorsage verwendet und aktualisiert.

Storage

Was sind Poling und Interrupts?

Storage

- **Poling:**
  - Perdiodisch I/O Status Register lesen
  - Für kleine Geräte nützlich, ansonsten schlecht für CPU Zeit
- **Interrupts:**
  - Controller unterbricht die CPU mit Interrupt, sobald Gerät bereit ist oder Fehler vorliegt

Storage

Was beeinflusst die Performance einer Festplatte?

Storage

Seek Time, Cache und Fragmentierung

Storage

Was macht Raid?

Storage

- Mehrere Kleine Festplatten anstatt einer großen
- Parallelität verbessert die Leistung
- Extra Platten für redundante Datenspeicherung

Storage

Wie funktionieren die verschiedenen Raid systeme?

Storage

- **RAID 1** ("Spiegel")
  - 2 Platten, 1:1-Kopie
  - ■ Sicher, schnell lesen
  - – Nur 50% Platz
- **RAID 2**
  - Bits + ECC (veraltet)
- **RAID 3**
  - Bytes + 1 Paritätsplatte
  - – Parität = Flaschenhals
- **RAID 5**
  - Blöcke + rotierende Parität
  - ■ Guter Mix (Platz/Sicherheit)
  - – Langsam schreiben

Storage

Woraus bestehen Festplatten?

Storage

Platten -> Tracks -> Sektoren welche durch eine Partitionstabelle (MBR oder GPT) bestimmt wird

Virtualisierung

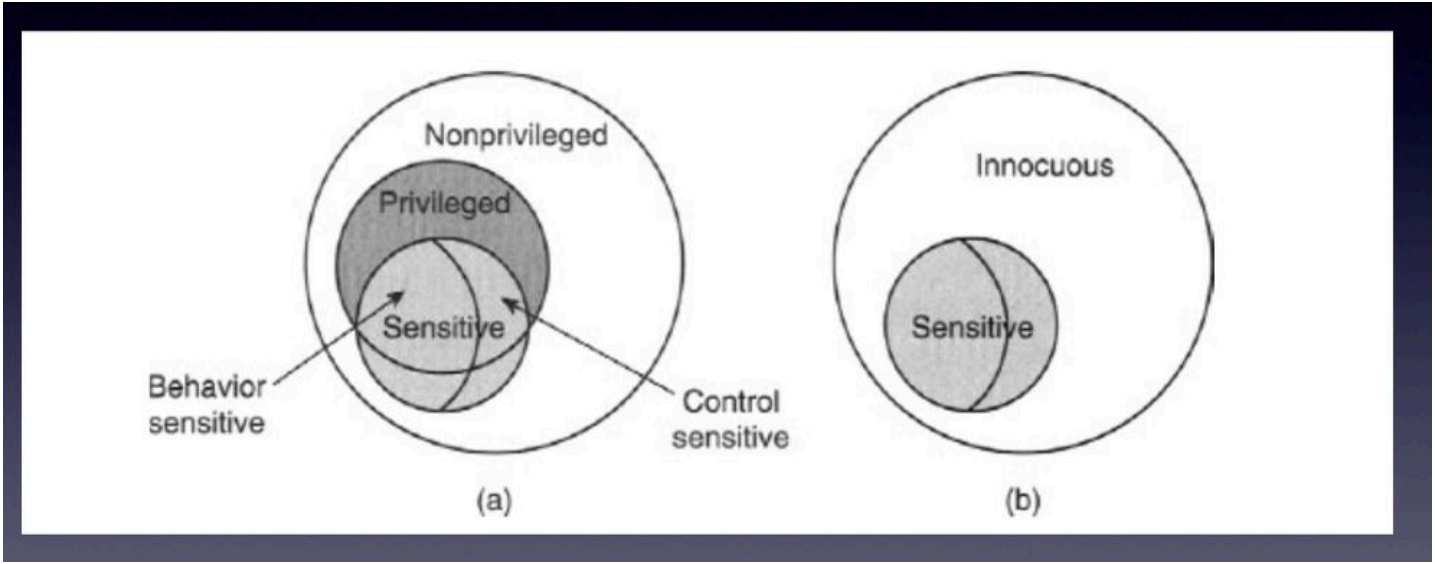
Nenne 3 eigenschaften der Interposition bei Virtualierung

Virtualisierung

Kompression
Verschlüsselung
Profiling
Binary Translation
Sciherheit z.Z. Virenschutz
Debugging

Virtualisierung

Erkläre folgendes Bild:



Virtualisierung

Zeigt die verschiedene Kategorien von Befehlen oder Operationen in einem Computersystem klassifizieren.
Hilft, die **Sicherheit** und **Stabilität** von Computersystemen zu gewährleisten.

1. Nicht-privilegiert (Nonprivileged)

- Keine besonderen Rechte nötig.
- Meist harmlos (Innocuous).

2. Privilegiert (Privileged)

- Erfordert Kernel-/Admin-Rechte.
- **Steuerungs-sensitiv**: Ändert Ressourcen (z. B. Speicher).
- **Verhaltens-sensitiv**: Ergebnis hängt von Konfiguration ab

3. Sensitiv

- Kombiniert steuerungs- und verhaltenssensitive Befehle.
- Kritisch für Systemstabilität/Sicherheit.

4. Harmlos (Innocuous)

- Kein Einfluss auf System, keine Privilegien nötig

Virtualisierung

Nenne 3 eigenschaften der Isolierung bei Virtualierung

Virtualisierung

Fehlerisolierung
Versionierung zur vermeidung von DLL-Hell
Performance Isolierung

Virtualisierung

Was ist der Virtual Machine Monitor?

Virtualisierung

Ein VMM ist eine Software/Hardware-Schicht, die mehrere virtuelle Maschinen (VMs) auf einer physischen Hardware steuert.

Funktion:

- Isoliert VMs voneinander (z. B. unterschiedliche Betriebssysteme auf einem Server).
- Verteilt Hardware-Ressourcen (CPU, RAM) an VMs.
- Erzwingt Sicherheitsregeln (kein direkter Zugriff auf physische Geräte).

Virtualisierung

Nenne 3 eigenschaften der Datenkapselung bei Virtualierung

Virtualisierung

- **Sicherheit:** Kein direkter Zugriff auf VM-internen Speicher.
- **Flexibilität:** VMs lassen sich pausieren/fortsetzen (z. B. via `virsh save`).
- **Skalierung:** Cloud-Dienste nutzen Kapselung für Massen-Hosting.

Virtualisierung

Was ist ein Container?

Virtualisierung

Ein
Container
ist eine
isoliere Laufzeitumgebung
, die eine Anwendung mit allen benötigten Abhängigkeiten (Bibliotheken, Konfiguration) verpackt.

Virtualisierung

Container vs VMs

Virtualisierung

- **Container** = "Lightweight-VM" für Apps mit gleichem OS-Kernel.
- **VM** = "Kompletter PC im PC" für maximale Isolation.

Virtualisierung

Nenne drei wichtige eigenschaften der Virualisierung mit erklärung.

Virtualisierung

1. Isolierug
   - Anwendungen laufen getrennt voneinander und können sich nicht gegenseitig beeinflussen
2. Interposition
   - Alle Operationen des Gastbetriebs laufen durch den Virtual Machine Monitor
3. Datenkapselung
   - Der Zustand der VM kann in einer Datei gespeichert werden

Virtualisierung

Wo werden Virtuelle Maschinen eingesetzt?

Virtualisierung

Cloud Computing: VMs ermöglichen die effiziente Bereitstellung von Ressourcen unter Wahrung der Isolierungsanforderungen (IaaS, PaaS, SaaS)