

## Organisation

Vorlesung  
Übung  
Prüfungen

## Einführung

Überblick  
Chancen und Limitierungen  
Herausforderungen



## ■ Verantwortliche

- Tobias Distler Raum 0.041 distler@cs.fau.de
- Jürgen Kleinöder Raum 0.043 jk@cs.fau.de

## ■ Termin

- Donnerstag, 12:15 – 13:45 Uhr
- Raum 0.031-113

## ■ Web-Seiten

- Skript: [http://www4.cs.fau.de/Lehre/WS12/V\\_MW/Vorlesung/](http://www4.cs.fau.de/Lehre/WS12/V_MW/Vorlesung/)
- Literatur: [http://www4.cs.fau.de/Lehre/WS12/V\\_MW/Literatur/](http://www4.cs.fau.de/Lehre/WS12/V_MW/Literatur/)

## ■ Rückmeldungen und Fragen

- Bitte Fragen stellen!
- Auf Fehler aufmerksam machen



## ■ Themen

- Überblick über Cloud Computing
- Grundlagen verteilter Programmierung mit Web-Services
- Virtualisierung als Basis für Cloud Computing
- Infrastructure as a Service (IaaS)
  - Amazon EC2
  - Eucalyptus
  - Windows Azure Storage
- Verteilte Datenspeicher für Cloud-Anwendungen
  - Google File System
  - Dynamo
- Verteilte Programmierung für datenintensive Cloud-Anwendungen
- Koordinierung von Cloud-Anwendungen
- Interoperabilität und Multi-Cloud Computing
- Fehlertoleranz im Kontext von Cloud Computing



## ■ Verantwortliche

- Tobias Distler Raum 0.041 distler@cs.fau.de
- Timo Hönig Raum 0.036 thoenig@cs.fau.de
- Johannes Behl Raum 0.041 behl@cs.fau.de
- Klaus Stengel Raum 0.041 stengel@cs.fau.de

## ■ Termine

- Tafelübung: Freitag, 12:15 – 13:45 Uhr, Raum 0.031-113 (**ab 26. 10.**)
- Rechnerübung: Freitag, 14:00 – 16:00 Uhr, Raum 00.156-113 (**ab 2. 11.**)

## ■ Web-Seite

- [http://www4.cs.fau.de/Lehre/WS12/V\\_MW/Uebung/](http://www4.cs.fau.de/Lehre/WS12/V_MW/Uebung/)

## ■ Anmeldung

- Web-Anmeldesystem *Waffel*
- <https://waffel.informatik.uni-erlangen.de>



- Tafel- und Rechnerübung
  - Ergänzende und vertiefende Informationen zur Vorlesung
  - Hilfestellungen zur Bearbeitung der Übungsaufgaben
  - Klärung von Fragen
  - Abgabe der Übungsaufgaben
- Themen
  - Entwicklung Cloud-basierter Web-Services
  - Einsatz einer privaten IaaS-Cloud (Eucalyptus)
  - Implementierung skalierbarer Algorithmen mittels Hadoop MapReduce
  - Koordinierung von verteilten Cloud-Anwendungen
  - Dynamische Skalierbarkeit von Cloud-basierten Diensten in Amazon EC2



- Informatik (Bachelor und Master)
  - Vertiefung „Verteilte Systeme und Betriebssysteme“
  - 5 ECTS- oder 7,5 ECTS-Modul
- Informations- und Kommunikationstechnik
  - Bachelor: „Wahlmodule aus EEI und INF“ (5 ECTS-Modul)
  - Master: „Wahlpflichtmodul aus INF“ (5 ECTS- oder 7,5 ECTS-Modul)
    - Eingebettete Systeme
    - Kommunikationsnetze
    - Übertragung und Mobilkommunikation
- Varianten
  - 5 ECTS: Vorlesung + Übung
    - Erfolgreiche Bearbeitung aller abzugebenden Übungsaufgaben
    - Mündliche Prüfung über Vorlesungs- und Übungsstoff (20 Minuten)
  - 7,5 ECTS: Vorlesung + erweiterte Übung
    - Erfolgreiche Bearbeitung aller abzugebenden Übungsaufgaben
    - Erfolgreiche Bearbeitung aller Zusatzaufgaben
    - Mündliche Prüfung über Vorlesungs- und Übungsstoff (30 Minuten)



## Überblick

### Organisation

Vorlesung

Übung

Prüfungen

### Einführung



Überblick

Chancen und Limitierungen

Herausforderungen

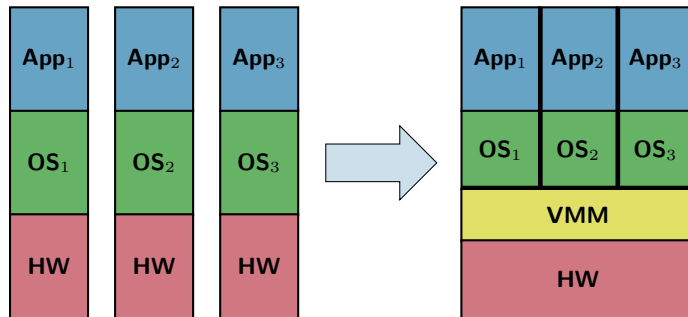


## Cloud Computing

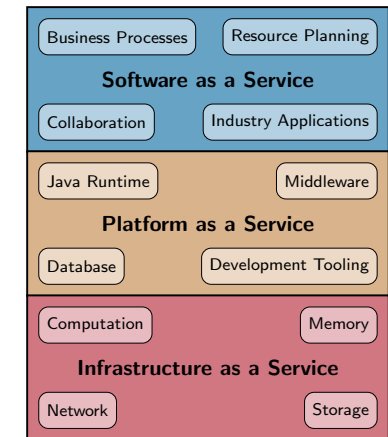
- Merkmale
  - Auslagerung von Diensten, Berechnungen und/oder Daten
  - Verfügbarkeit scheinbar unbegrenzter Ressourcen
  - Einfacher universeller Zugriff
  - Schnelle dynamische Skalierbarkeit
- Grundlagen
  - Hochskalierbare verteilte Infrastrukturen auf Provider-Seite
  - Leistungsfähige Netzwerkanbindung auf Client-Seite
  - Geringe Kosten für Speicherplatz
- Literatur
  -  Mache Creeger  
**Cloud Computing: An Overview**  
*Queue – Distributed Computing*, 7(5), 2009.
  -  Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph et al.  
**A View of Cloud Computing**  
*Communications of the ACM*, 53(4):50–58, 2010.



- Web-Services
  - Sprachunabhängige Basis für entfernte Kommunikation
  - Bereitstellung von Diensten in der Cloud
  - Schnittstelle zur Cloud-Konfiguration
- Virtualisierung
  - Paralleler Betrieb mehrerer *virtueller Maschinen* auf einem Rechner
  - Isolation von Nutzern
  - Realisierung auf unterschiedlichen Ebenen: System, Prozess,...



- Kategorien
  - Software as a Service (SaaS)
    - Bereitstellung vom Endnutzer verwendeter Dienste
    - Beispiel: Google Docs
  - Platform as a Service (PaaS)
    - Bereitstellung von Middleware zur Implementierung komplexer Dienste
    - Beispiel: Google AppEngine
  - Infrastructure as a Service (IaaS)
    - Bereitstellung von Rechen- und Speicherinfrastruktur
    - Beispiel: Amazon EC2
- In der Praxis
  - Oftmals als Schichten aufeinander aufbauend
  - Grenzen zwischen Kategorien fließend



- Öffentliche Cloud (*Public Cloud*)
  - Große Unternehmen (z. B. Amazon, Microsoft, Google) stellen anderen Firmen einen Teil ihrer Infrastruktur zur Verfügung
  - Cloud-Nutzer müssen selbst vergleichsweise wenige Ressourcen vorhalten
- Private Cloud
  - Nutzung der bereits im eigenen Unternehmen vorhandenen Infrastruktur
  - Einsatz von Virtualisierung zur flexiblen Verwaltung von Ressourcen
- Hybride Cloud
  - Kombination aus privater und öffentlicher Cloud
  - Mögliche Aufteilung
    - Kritische Daten verbleiben im privaten Teil der Cloud
    - Öffentliche Cloud vor allem zur Deckung von Bedarfsspitzen
- Multi Clouds / Cloud-of-Clouds
  - Parallele Nutzung verschiedener öffentlicher Clouds
  - Absicherung gegen den Ausfall eines Cloud-Anbieters

- Cloud-Nutzer
  - Keine eigenen physischen Rechner
    - Geringerer Aufwand für Administration
    - Keine Reparaturkosten
  - Abrechnungsmodell: *Pay-as-you-go*
    - Kosten orientieren sich an tatsächlichem Ressourcenverbrauch
    - Feingranulare Abrechnung [Beispiele: Virtuelle Maschine: pro Stunde, Netzwerk: pro Megabyte]
    - Konsequenzen (Beispiele)
      - \* 1000 virtuellen Maschinen eine Stunde lang zu betreiben kostet genauso viel wie eine virtuelle Maschine 1000 Stunden lang zu betreiben
      - \* Keine Kosten für On-Demand-Dienste während der Standby-Phase, zum Beispiel im Rahmen von Garantieleistungen
  - In vielen Fällen: Verbesserte Ausfallsicherung
- Cloud-Anbieter
  - Vermietung überschüssiger Kapazitäten
  - Virtualisierung: Bereitstellung quasi unbegrenzter Ressourcen für Nutzer

## Datenzentren

- Einsatz von *Commodity*-Hardware
  - Keine Spezialanfertigungen, sondern Hardware von der Stange
  - Vorteil: Günstige Einkaufspreise aufgrund großer Stückzahlen
  - Nachteile
    - Ausfälle werden zum Regelfall
    - Kompatibilitätsprobleme aufgrund heterogener Hardware
- Server-Konsolidierung durch Virtualisierung
  - Zusammenlegung von virtuellen Maschinen verschiedener Cloud-Nutzer auf dem selben physischen Rechner
  - Vorteile
    - Höhere Auslastung einzelner Rechner  
[Ohne Virtualisierung: 2-3%, mit Virtualisierung: bis zu 80%. [Creeger]]
    - Kostenersparnis durch geringeren Platzbedarf
  - Nachteil: Optimale Isolation ist nicht immer erreichbar
    - Sicherheitsaspekt
    - *Performance Isolation*



## Skalierbarkeit

- Problem von Firmen: Abschätzung der Auslastung ihrer für Endnutzer angebotenen Dienste und Bereitstellung entsprechender Ressourcen
  - Lastentwicklung eventuell unbekannt
  - Ungünstiges Verhältnis zwischen Spitzen- und Durchschnittslast
  - Starke Lastschwankungen
    - Über den Tag verteilt
    - Übers Jahr verteilt
- Mögliche Konsequenzen
  - Bereitstellung von zu wenigen Ressourcen (*Underprovisioning*)
  - Bereitstellung von zu vielen Ressourcen (*Overprovisioning*)
- Vorteile durch Cloud Computing
  - Verfügbarkeit neuer virtueller Maschinen im Minutenbereich
  - Dynamische Skalierbarkeit in beide Richtungen
  - Kosten orientieren sich am tatsächlichen Ressourcenverbrauch



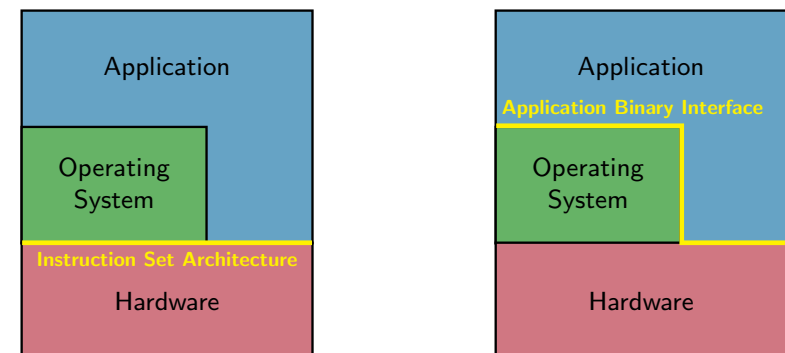
## Limitierungen und offene Fragen

- Technische Limitierungen
  - Nicht jede Anwendung lässt sich beliebig skalieren (z. B. Datenbanken)
  - Ineffizienter Transfer großer Datenmengen in die bzw. aus der Cloud  
[Amazon bietet zum Beispiel die Möglichkeit Festplatten einzusenden: <http://aws.amazon.com/importexport/>]
  - Instabile bzw. unvorhersehbare Performanz von schwer transparent zu isolierenden Operationen (z. B. Festplattenzugriffen)
  - Beschränkte Verfügbarkeitsgarantien durch Cloud-Anbieter
- Weiterführende Aspekte
  - Vertraulichkeit der Daten
  - Rechtliche Fragen (Beispiele)
    - Dürfen medizinische Daten in einer öffentlichen Cloud verarbeitet werden?
    - Werden gesetzliche Bestimmungen zum Speicherort von Daten eingehalten?
  - „*Vendor Lock-In*“-Problem
    - Starke Abhängigkeit von einem einzelnen Cloud-Anbieter
    - Erschwerter Anbieterwechsel
    - Gründe: fehlende Standards, aufwendiger Datentransfer



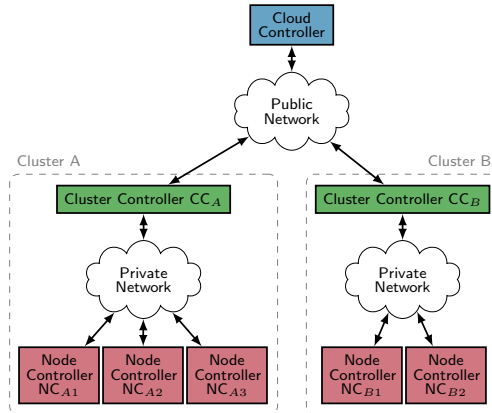
## Wie lässt sich Virtualisierung praktikabel realisieren?

- Anforderungen an ein virtualisiertes System
  - Äquivalenz
  - Ressourcenkontrolle
  - Effizienz
- Virtualisierungsebenen
  - Systemvirtualisierung: Virtualisierung der *Instruction Set Architecture*
  - Prozessvirtualisierung: Virtualisierung des *Application Binary Interface*



## Wie wird die eigene Infrastruktur für andere nutzbar?

### ■ Aufbau einer Infrastruktur-Cloud



### ■ Aufgabenbereiche

- Verwaltung von physischen Maschinen
- Verwaltung und Platzierung von virtuellen Maschinen
- Anbindung an Datenspeicher



## Wie lassen sich große Datenmengen verwalten?

### ■ Ansatz

- Speziell auf die jeweiligen Anforderungen zugeschnittene Systeme
- Enge Verzahnung mit der Anwendung

### ■ Beispiel: Google

- Anforderungen
  - Sehr große Dateien
  - Hauptsächlich sequentielle Schreibzugriffe, kaum Modifikationen
- *Google File System*
  - Kein Dateisystem im klassischen Sinne
  - Optimierte Auslastung der Netzwerkverbindungen

### ■ Beispiel: Amazon

- Anforderungen
  - Große Anzahl an vergleichsweise kleinen Datensätzen
  - Hohe Verfügbarkeit
- *Amazon Dynamo*
  - Replizierter Datenspeicher für Schlüssel-Wert-Paare
  - Abgeschwächte Konsistenzgarantien



## Wie lassen sich große Datenmengen verarbeiten?

### ■ Beispiel: Google, Yahoo,...

- Anforderungen
  - Parallele Nutzung einer großen Anzahl von Rechnern
  - Einfache Realisierung von Anwendungen
- *MapReduce*
  - Framework übernimmt Verteilung der Anwendung
  - Programmierer implementiert zwei Methoden
    - \* Map: Abbildung der Eingabedaten auf Schlüssel-Wert-Paare
    - \* Reduce: Zusammenführung der von Map erzeugten Schlüssel-Wert-Paare

### ■ Koordinierung und Konfiguration verteilter Anwendungen

- Anforderungen
  - Abstimmung zwischen einer großen Anzahl von Prozessen
  - Ausfallsichere Verwaltung von Konfigurationsinformationen
- Beispiel: *Chubby* (Google)
  - Bereitstellung als externer Koordinierungsdienst
  - Generische Schnittstelle zur Implementierung komplexer Abstraktionen

