FRIEDRICH-SCHILLER-
UNIVERSITÄT
JENA

# Datenbanksysteme II

Prof. Dr. Viktor Leis

Professur für Datenbanken und Informationssysteme

# Cloud Computing

# Cloud Computing

- Was?
  - Rechenleistung wird nach Bedarf bezahlt
    (Analogie: Stromnetz)
  - Möglichkeit Rechner je nach Arbeitslast hinzuzufügen oder abzuschalten
- Warum?
  - Kosten (ökonomische Skaleneffekte, total cost of ownership)
  - Skalierbarkeit
  - Elastizität
- Folien nach Peter Boncz

# Service Provider: Die Großen Drei

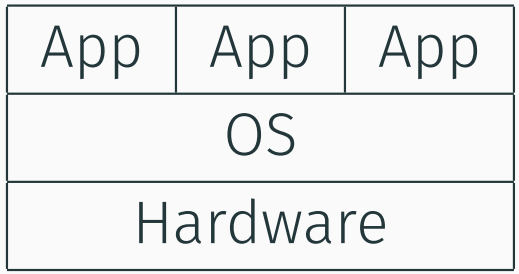
- Amazon Web Services
- Microsoft Azure
- Google Cloud Platform

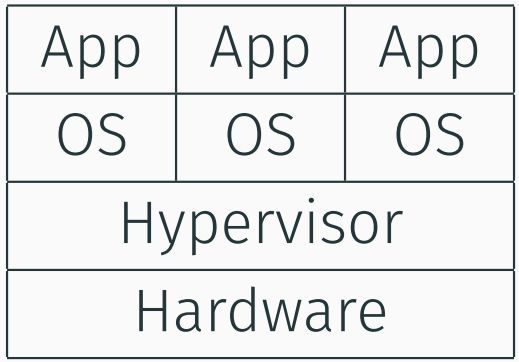
# Everything as a Service

- Infrastructure as a Service: Vermietung von Hardware (Amazon EC2)
- Platform as a Service: API (Google App Engine)
- Software as a Service: Webapp (Gmail)

# Zugrundeliegende Technologie: Virtualisierung

- Hypervisor (z.B. Xen, KVM) erlauben Isolation/Virtualisierung mehrerer Rechner auf einem physischen Rechner
- Hardwareunterstützung: Intel VT-x, AMD-V



# Beispiel: Amazon Web Services (AWS)

Infrastructure:

- EC2: virtual machines
- S3: distributed storage
- S3 Glacier: backup storage
- ...

Platform:

- RDS: PostgreSQL, MySQL, Oracle, SQL Server
- Aurora: OLTP system (originally based on MySQL)
- DynamoDB: Key/Value store
- Redshift: OLAP system (originally based on PostgreSQL)
- ...

# AWS Terminologie

- instance ≈ virtueller Rechner
- vCPU ≈ HyperThread
- instance storage ≈ lokale Festplatte/SSD
- availability zone ≈ Rechenzentrum
- region ≈ geografisches Gebiet

# Amazon EC2

großes Angebot an (virtualisierten) Rechnern

- ca. 20 global verteilte Rechenzentren (USA, Europa, Asien)
- die meisten Instanzen werden geteilt
- Konfiguration: Anzahl Kerne, Arbeitsspeicher, SSDs, Netzwerk, GPUs, FPGAs, ...
- Preis von $0.005 bis $30 pro Stunde
- Netzwerk nur innerhalb availability zone umsonst

`https://ec2instances.info/`

`https://aws.amazon.com/ec2/instance-types/`

`https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Concepts.RegionsAndAvailabilityZones.html`

# EC2 Preismodell

- on-demand: sekundengenaue Abrechnung
- reserved: Reservierung für 1 oder 3 Jahre, 40-60% Rabatt
- spot: variable Kosten (oft nur 30%), Unterbrechung jederzeit möglich

# Amazon S3

S3 ist ein verteiltes Dateisystem:

- Dateien (“objects”) haben einen eindeutigen Schlüssel
- Zugriff über HTTP: GET, POST, DELETE, LIST
- Skalierbarkeit durch Verteilung über mehrere Rechner (lokale Platten von sehr vielen Instanzen bilden die Basis)
- Fehlertoleranz durch redundante Speicherung
- schwache Konsistenz: veraltete Daten können sichtbar sein (“eventual consistency”)
- günstige Version für Backups: Glacier

`https://aws.amazon.com/s3/pricing/`

`https://aws.amazon.com/glacier/pricing/`

# Elastic Block Service (EBS)

EBS ist eine virtuelle Festplatte

- Block-basiert (in S3 ist Datei-basiert)
- kann als Dateisystem eingebunden werden (mount)
- im Gegensatz zu lokalen Platten (“instance storage”), ist EBS persistent
- teurer als S3

`https://aws.amazon.com/ebs/pricing/`

# Nachteile der Cloud

- Datenschutz
- Virtualisierung kann Performance kosten
- Performance kann schwanken und vom Verhalten anderer Nutzer (“noisy neighbours”) abhängen
- Kosten können hoch sein
- Abhängigkeit vom Cloud Provider

# Verteilte Anwendungsentwicklung

- das Versprechen der Cloud ist Skalierbarkeit (über viele Rechner) und Elastizität
- das schwierige hierbei ist nicht die Hardware, sondern die Software
- Herausforderungen:
  - Parallelisierung
  - Verteilung der Daten
  - Kommunikation zwischen Rechnern
  - Ausfall von Rechnern

# Technologien für Verteilte Anwendungsentwicklung

- Verteilte Dateisysteme: S3, HDFS
- Key/Value Stores: nicht-relationale Datenbanksysteme (keine Transaktionen, kein Schema, kein SQL aber dafür skalierbar)
- Map/Reduce: Programmiermodell für verteilte Anwendungen (stirbt aber langsam wieder aus)
- Spark: generische Datenverarbeitungsplattform (DataFrame ähnlich zu Relation)

# Cloud: Zusammenfassung

- man kann davon ausgehen, dass in Zukunft immer weniger Organisationen Hardware selber betreiben
- Unternehmen wollen
  - nur so viel zahlen, wie sie wirklich benutzen (z.B. pro Zugriff) und
  - der Service soll beliebt skalieren und
  - die Zugriffszeiten sollen immer niedrig sein (niedrige “tail latency”)
- dies stellt die Softwareindustrie vor großen Herausforderungen
- viel ungelöste (Forschungs-)Probleme (gerade im Datenbankbereich)