

## Multi-Cloud Computing

Motivation

Architekturen

Deltacloud

Redundant Array of Cloud Storage (RACS)

Zusammenfassung



- Typische Vorgehensweise bei der Migration eines Diensts in die Cloud
  - Auswahl eines Cloud-Anbieters anhand bestimmter Kriterien (Beispiele)
    - Kostenabschätzung
    - Leistungsgarantien (SLAs)
  - Anpassung des Diensts auf die spezifischen Gegebenheit einer Cloud
  - Probleme
    - Preismodell, Leistungsgarantien,... eines Anbieters können sich ändern
    - Ein Anbieterwechsel ist in der Regel sehr aufwendig
- Mögliche Gründe für erschwerten Anbieterwechsel (Beispiele)
  - Technisch nicht kompatible Cloud-Systeme
    - (Noch) Keine einheitlichen Standards im Einsatz
    - Nutzung anwenderspezifischer Funktionalität
  - Hoher zeitlicher/finanzieller Aufwand [z. B. falls Transfer großer Datenmengen erforderlich.]
- Herausforderungen
  - Wie lässt sich der Aufwand für einen Anbieterwechsel reduzieren?
  - Wie kann ein Dienst auf Basis mehrerer Clouds bereitgestellt werden?




## Multi-Cloud Computing

- Alternativ verwendete Begriffe
  - Cloud-of-Clouds
  - Federated Cloud
  - Inter-Cloud
  - ...
- Gründe für den Betrieb einer Multi-Cloud (Beispiele)
  - Abschwächung des „Vendor Lock-In“-Problems
  - Nutzung der Vorzüge unterschiedlicher Cloud-Anbieter
    - Spezifische Funktionalität
    - Geografische Lage von Datenzentren
    - ...
  - Steigerung der Zuverlässigkeit
- Grundlegende Entwicklungskonzepte für Multi-Cloud-Anwendungen
  - Vollständig Cloud-agnostischer Ansatz
  - Einsatz von anbiertergewahren Komponenten



## Multi-Cloud-Architekturen

- Hybride Architektur
  - Lose gekoppelte Kombination aus privater und öffentlicher Cloud
  - Privater Teil der Cloud
    - Virtualisierte Infrastruktur im firmen-/institutionseigenen Datenzentrum
    - Volle Kontrolle über die bereitgestellten Ressourcen
  - Öffentlicher Teil der Cloud
    - Angemietete Ressourcen in einer oder mehreren öffentlichen Clouds
    - Eingeschränkte Kontrolle über die bereitgestellten Ressourcen
  - Mögliches Anwendungsszenario
    - Ausschließliche Nutzung des privaten Cloud-Teils bei Normalauslastung
    - Bewältigung von Lastspitzen durch Erweiterung des öffentlichen Cloud-Teils
- Literatur
  -  Rafael Moreno-Vozmediano, Rubén S. Montero, and Ignacio M. Llorente **laaS cloud architecture: From virtualized datacenters to federated cloud infrastructures** *Computer*, 45(12):65–72, 2012.



- Broker-Architektur
  - Kombinierte Nutzung unterschiedlicher öffentlicher Clouds
  - *Broker*
    - (Transparente) Proxy-Komponente für Zugriff auf unterschiedliche Clouds
    - Auswahl der öffentlichen Cloud, auf der eine Client-Anfrage ausgeführt wird
    - Mögliche Entscheidungskriterien: Kosten, Performanz,...
  - Beispiele
    - Deltacloud
    - Redundant Array of Cloud Storage (RACS)
- Aggregierte Architektur
  - Direkte Kooperation mehrerer privater oder öffentlicher Clouds
  - Bereitstellung gemeinschaftlicher Ressourcen
- Mehrschichtarchitektur
  - Hierarchische Komposition mehrerer privater oder öffentlicher Clouds
  - Abstraktion einer einzelnen großen Cloud

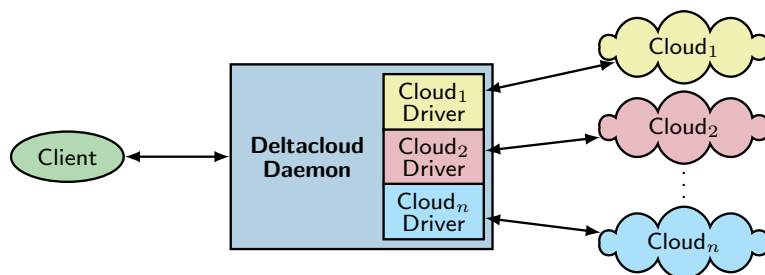


- Problem
  - Keine standardisierten Schnittstellen für IaaS-Clouds
  - Client-APIs variieren zwischen Cloud-Anbietern
 → Aufwendige Portierung einer Anwendung bei Wechsel des Cloud-Anbieters
- Deltacloud
  - Apache-Projekt (Open-Source)
  - Proxy-basierter Ansatz
  - Bereitstellung einer Cloud-Anbieter-unabhängigen REST-Schnittstelle
  - Unterstützte Plattformen
    - Virtuelle Maschinen: Amazon EC2, Eucalyptus, Rackspace, OpenStack,...
    - Datenspeicher: Amazon S3, Windows Azure Storage, Google Storage,...
- Web-Seite
  - [Deltacloud](http://deltacloud.apache.org/)
  - <http://deltacloud.apache.org/>



## Architektur

- Deltacloud-Daemon
  - Bereitstellung einer einheitlichen Web-Service-Schnittstelle für Clients
  - Platzierungsvarianten
    - Eigene Instanz auf dem Rechner oder im Datenzentrum des Client
    - Öffentliche Deltacloud-Instanzen
- Cloud-spezifische Treiber
  - Übersetzung der Client-Anfragen in Zugriffe auf die jeweilige IaaS-Cloud
  - Nutzung der nativen Schnittstelle eines Cloud-Anbieters



## Schnittstelle

- Von spezifischen Clouds abstrahierte API
  - Nicht alle der Abstraktionen sind für alle Cloud-Anbieter verfügbar
  - Zugriff über REST
- Cloud-Anbieter-unabhängige Abstraktionen (Beispiele)

Datenstruktur	Beschreibung
Realm	Beschränkter Ressourcenbereich [vgl. Availability-Zone in Amazon EC2]
Image	Abbild einer virtuellen Maschine (VM)
Instance	Eine sich in der Ausführung befindliche VM
Instance State	Zustand einer VM [z.B. start, running, shutting-down]
Load Balancer	Konfiguration für Lastbalancierung zwischen VMs
Firewall	Regeln für Netzwerkzugriffe auf eine VM
Blob	Speichereinheit für Binärdaten [vgl. Objekt in Amazon S3]
Bucket	Container für Blobs

- Verfügbarer Methodenumfang (Beispiele)
  - Erzeugen, Starten und Herunterfahren von virtuellen Maschinen
  - Lese- und Schreibzugriff auf Cloud-Datenspeicher



- Naiver Ansatz für redundante Speicherung von Daten
  - Vollständige Replikation der Daten über mehrere Clouds
  - Nachteile
    - Hohe Kosten für Datenspeicherung und -transfer
    - Hinzufügen eines weiteren Anbieters erfordert Umkopieren aller Daten
- Redundant Array of Cloud Storage (RACS)
  - Fokus auf Anbieterwechsel aus wirtschaftlichen Gründen
    - Wechsel kündigt sich vergleichsweise lange im Voraus an
    - Einbindung möglichst vieler Anbieter sinnvoll
  - Vorbild: Redundant Array of Independent Disks (RAID)
  - Redundante Speicherung von Daten mittels *Erasure-Codes*
- Literatur
  - 📄 [Hussam Abu-Libdeh, Lonnie Princehouse, and Hakim Weatherspoon](#)  
**RACS: A case for cloud storage diversity**  
*Proceedings of the 1st Symposium on Cloud Computing (SoCC '10)*, pages 229–240, 2010.

- Allgemeines
  - Verfahren zur Vorwärtsfehlerkorrektur
  - Einsatzbereiche (Beispiele): Datenübertragung, -speicherung
  - Beispiel *Reed-Solomon-Codes*: RAID 6, CDs,...
- Grundlegender Ansatz
  - Zerlegung eines zu speichernden Objekts  $O$  in  $m$  gleichgroße Teile
  - Berechnung von  $n > m$  Fragmenten basierend auf den  $m$  Teilen
  - Besondere Eigenschaft des Abbildungsverfahrens: Die originalen  $m$  Teile lassen sich aus  $m$  beliebigen der  $n$  Fragmente rekonstruieren

→ Kein Datenverlust solange höchstens  $n - m$  Fragmente verloren gehen
- Speicherbedarf abhängig von  $m$  und  $n$ 
  - Faktor  $f = \frac{n}{m}$  im Vergleich zur nicht-redundanten Speicherung
  - Beispiel: Verteilung auf 7 Anbieter, Tolerierung eines Weg- bzw. Ausfalls
    - $m = 6, n = 7 \Rightarrow f \approx 1.17$
    - Vergleich:  $f = 7$  [7 Anbieter] bzw.  $f = 2$  [Tolerierung eines Ausfalls] bei naivem Ansatz

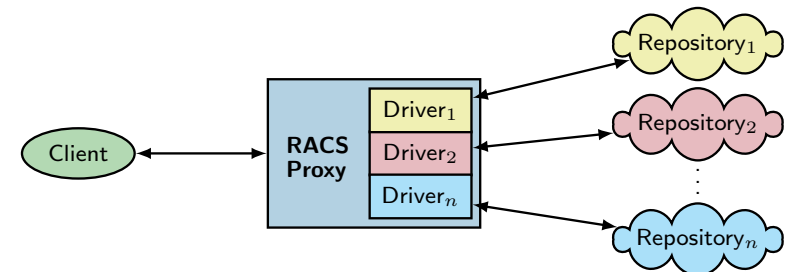
## Schnittstelle

- Datenmodell
  - Anlehnung an das Datenmodell von Amazon S3
  - Verwaltung von Schlüssel-Wert-Paaren
  - Datenstrukturen
    - Verzeichnisse: *Buckets*
    - Binärdateien: *Objekte*
- Schnittstelle
  - Ziel: Generische, vom einzelnen Anbieter unabhängige Schnittstelle
  - Verfügbare Operationen

Operation	Beschreibung
create()	Anlegen eines Bucket
put()	Hinzufügen eines Objekts zu einem Bucket
get()	Auslesen eines Objekts
delete()	Löschen eines Objekts bzw. Bucket
list()	Auslesen von Metadaten eines Bucket

## Architektur

- Basisarchitektur
  - Client-Anwendung [z. B. unmodifizierter Amazon-S3-Client]
  - RACS-Proxy
    - Übersetzung von Client-Aufrufen in Anfragen an verschiedene Clouds
    - Spezifischer Adapter für jeden Cloud-Speicher-Anbieter
  - Repositories: Datenspeicher bei unterschiedlichen Cloud-Anbietern



- Verteiltes RACS
  - Erweiterung der Basisarchitektur um weitere RACS-Proxies
  - Koordinierung zwischen RACS-Proxies mittels ZooKeeper

- Bearbeitung von Anfragen
  - Schreibenanfragen
    - Berechnung von  $n$  Objektfragmenten mittels Erasure-Codes
    - Verteilung der Objektfragmente auf  $n$  Cloud-Speicher-Anbieter
    - Vollständige Replikation der Metadaten auf alle  $n$  Repositories
  - `get()`-Anfragen
    - Holen von  $m$  Objektfragmenten aus verschiedenen Repositories
    - Rekonstruktion des angeforderten Objekts
  - `list()`-Anfragen
    - Auslesen der Metadaten aus einem beliebigen Repository
    - Vollständige Replikation von Metadaten → einzelne Operation reicht aus
- Optimierte Auswahl von Repositories mittels *Policy Hints*
  - Vorgaben, welche Repositories (z. B. bei Leseanfragen) zu bevorzugen sind
  - Beispiele für Kriterien
    - Kosten für Datentransfer
    - Geografisch bedingte Latenzen



- „Vendor Lock-In“-Problem
  - Cloud-Anbieter-Wechsel sind in der Regel sehr aufwendig
  - Technische und finanzielle Hürden bei Wechsel des Anbieters
- Multi-Cloud-Architekturen
  - Hybride Architektur
  - Broker-Architektur
  - Aggregierte Architektur
  - Mehrschichtarchitektur
- Deltacloud
  - Einheitliche Schnittstelle für Client-Anwendungen
  - Cloud-spezifische Treiber
- Redundant Array of Cloud Storage (RACS)
  - Fokus auf Anbieterwechsel aus wirtschaftlichen Gründen
  - Proxy-basierter Ansatz
  - Redundante Speicherung von Daten mittels Erasure Codes

