

Cloud-Infrastrukturen

Motivation

Eucalyptus

Software-definierte Netzwerke

Zusammenfassung



Infrastructure as a Service (IaaS)

- Bereitstellung von Hardware-nahen Ressourcen
 - Virtuelle Maschinen auf Systemebene
 - Zuverlässiger und hochverfügbarer Datenspeicher
- Angestrebtes Preismodell: *Pay-as-you-go*
 - Abrechnung nach tatsächlich verbrauchten Ressourcen
 - Keine Fixkosten für Anschaffung, Betrieb und Wartung

→ Kosten orientieren sich im Optimalfall am Gewinn
- Dynamische Skalierbarkeit in beide Richtungen
 - Hinzufügen weiterer virtueller Maschinen bei Bedarfsspitzen
 - Herunterfahren von virtuellen Maschinen bei zu geringer Auslastung

■ Literatur



Peter Sempolinski and Douglas Thain

A comparison and critique of Eucalyptus, OpenNebula and Nimbus
*Proceedings of the 2nd International Conference on Cloud Computing
Technology and Science (CloudCom '10), S. 417–426, 2010.*



- Kompromiss bei der Platzierung von virtuellen Maschinen
 - Viele virtuelle Maschinen auf demselben Rechner → **hohe Auslastung**
 - Möglichst gleichmäßige Aufteilung der virtuellen Maschinen auf die vorhandenen Rechner → **geringe Beeinflussung** der VMs untereinander
 - In geringer geographischer Nähe zum Nutzer → **niedrige Latenz**
 - Geographisch weit entfernt von anderen virtuellen Maschinen derselben Anwendung → **hohe Ausfallsicherheit** des gesamten Dienstes
- Im Folgenden betrachtet
 - Wie kann eine weltumspannende Cloud-Infrastruktur mit Datenzentren auf verschiedenen Kontinenten realisiert werden?
 - Wie lässt sich die durch Virtualisierung erzielte Isolation auf Netzwerke von virtuellen Maschinen ausdehnen?
 - Wie läuft die Cloud-interne Kommunikation zwischen Datenzentren ab?



■ Motivation

- Einsatz von proprietären Implementierungen in kommerziellen IaaS-Clouds
- Kaum Informationen über den Aufbau solcher Systeme vorhanden
- Beschränkte Zugangsmöglichkeiten für Forscher

■ Eucalyptus

- Framework zur Verwaltung privater bzw. hybrider Clouds
- Zielgruppe: Universitäten und kleinere Firmen
- Anlehnung an Amazon EC2 bzw. Amazon S3
 - eucatools für Interaktion mit dem Framework
 - Client-Schnittstelle für Datenspeichersystem

■ Literatur



Daniel Nurmi, Rich Wolski, Chris Grzegorzcyk, Graziano Obertelli et al.

The Eucalyptus open-source cloud-computing system

Proceedings of the 9th International Symposium on Cluster Computing and the Grid (CCGrid '09), S. 124–131, 2009.



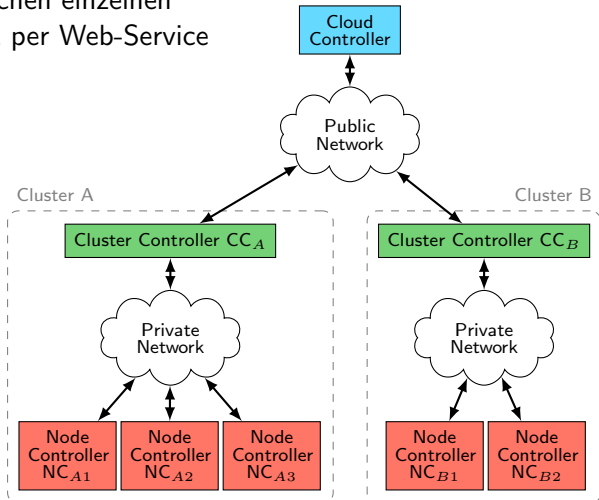
- Kommunikation zwischen einzelnen Komponenten erfolgt per Web-Service

- Controller-Hierarchie

- Cloud
- Cluster
- Node

- *Walrus*

- Datenspeicher
- Archiv für Images virtueller Maschinen
- Zugriff von inner- und außerhalb der Cloud möglich



■ *Cloud Controller*

- Zuständigkeitsbereich: komplette Eucalyptus-Cloud
- Schnittstelle zum Cloud-Nutzer bzw. -Administrator
 - Authentifizierung von Nutzern
 - Verwaltung von virtuellen Maschinen

■ *Cluster Controller*

- Zuständigkeitsbereich: Gruppe von Rechnern [Ausgangspunkt: Region in Amazon EC2]
- Bearbeitung von Anfragen des Cloud Controller
 - Auswahl der Node Controller für den Start einer virtuellen Maschine
 - Analyse der Kapazitäten für bestimmte VM-Typen
- Konfiguration und Überwachung von *Virtual Network Overlays*

■ *Node Controller*

- Zuständigkeitsbereich: (einzelner) lokaler Rechner
- Bearbeitung von Anfragen des zugehörigen Cluster Controller
 - Starten und Stoppen virtueller Maschinen
 - Berichte über Zustände lokaler virtueller Maschinen
 - Übersicht über Ressourcen (z. B. Anzahl an CPUs, freier Festplattenspeicher)



Start einer virtuellen Maschine

- Cloud Controller
 - Empfang einer Anfrage vom Nutzer
 - Überprüfung der Verfügbarkeit von Ressourcen
 - Reservierung der für die VM benötigten Ressourcen
 - Senden einer Anweisung an den Cluster Controller die VM zu starten
 - Nach Bestätigung: Aktualisierung der Ressourceninformationen
 - Cluster Controller
 - Auswahl des Rechners, auf dem die VM gestartet werden soll
 - Anwendung der *First-Fit*-Strategie
 - Node Controller
 - Bereitstellung des VM-Image auf dem Zielrechner (Varianten)
 - Transfer aus dem Image-Archiv von Walrus
 - Verfügbarkeit im lokalen Image-Cache
 - Konfiguration des Netzwerks (zusammen mit dem Cluster Controller)
 - Anweisung an den VMM das VM-Image zu booten
- Nutzer kann per `ssh` auf seine virtuelle Maschine zugreifen



- Anforderungen
 - **Isolation:** Eine VM eines Nutzers muss mit anderen VMs desselben Nutzers kommunizieren können, jedoch nicht mit VMs anderer Nutzer
 - **Erreichbarkeit:** Mindestens eine virtuelle Maschine jedes Nutzers muss von außerhalb der Cloud erreichbar sein
- Umsetzung mittels *Virtual Network Overlays*
 - Realisierung der Isolation
 - Einrichtung eines separaten virtuellen Netzwerks (VLAN) für jeden Nutzer
 - Jedes virtuelle Netzwerk verwendet ein eigenes IP-Subnetz
 - Cluster Controller
 - * Bei Bedarf: Routing zwischen IP-Subnetzen
 - * Isolation durch Firewall-Regeln
 - TCP- oder UDP-Tunnel zwischen verschiedenen Clustern
 - Einfluss auf Erreichbarkeit
 - Verwendung privater IP-Adressen → VMs von außen nicht zugänglich
 - Bei Bedarf Adressumsetzung von öffentlichen auf private IP-Adressen



- Schnittstelle: Orientierung an Amazon S3
 - Zugriff per REST oder SOAP
 - Verwaltung von Daten in *Objekten* und *Buckets*
 - Verwendung von Amazon-S3-Tools möglich
- Archiv für VM-Images virtueller Maschinen
 - Bestandteile eines VM-Image
 - Root Filesystem
 - Kernel Image
 - Ramdisk Image
 - Ablegen eines VM-Image
 - Komprimierung, Verschlüsselung und Aufteilung
 - Beschreibung des Image in einem *Manifest*
 - Bearbeitung von Node-Controller-Anfragen
 - Zusammenfügen, Verifizieren, Entschlüsselung und Entpacken
 - Transfer des Image zum Node Controller
 - Cache für entschlüsselte VM-Images



■ Motivation

- Plattform für Experimente mit neuen Netzwerkprotokollen
- Einheitlich programmierbare Netzwerkinfrastruktur
- Trennung zwischen Steuerlogik und eigentlicher Netzwerk-Hardware

■ OpenFlow

- Zentraler Begriff: *Flow*
 - Abstraktion eines Stroms von Netzwerkpaketen
 - Beispiele: Alle Pakete derselben TCP-Verbindung, Ursprungs-/Zieladresse,...
- Bestandteile
 - Switch mit von außen programmierbarer *Flow-Tabelle*
 - *Controller* zur Steuerung von Switches mittels Einträgen in Flow-Tabellen
 - OpenFlow-Protokoll zur Kommunikation zwischen Switch und Controller

■ Literatur



Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar et al.
OpenFlow: Enabling innovation in campus networks
SIGCOMM Computer Communication Review, 38(2):69–74, 2008.



- **Bestandteile eines Eintrags in der Flow-Tabelle**
 - **Paket-Header**
 - Maske der für den Flow charakteristischen Eigenschaften
 - Beispiele: {Ethernet,IP,TCP}-Ursprungs-/Zieladressen
 - **Auszuführende Aktion**
 - Weiterleitung des Pakets an einen bestimmten Port
 - Kapselung und Weiterleitung des Pakets an den Controller
 - Verwerfen des Pakets
 - **Statistiken**
 - Anzahl der Pakete und Bytes pro Flow
 - Empfangszeitstempel des neuesten Pakets eines Flow

- **Grundlegende Verarbeitungsschritte**
 1. Empfang eines Netzwerkpakets
 2. Suche nach einem zu dem Paket passenden Eintrag in der Flow-Tabelle
 3. Falls ein solcher Eintrag existiert: Ausführung der entsprechenden Aktion



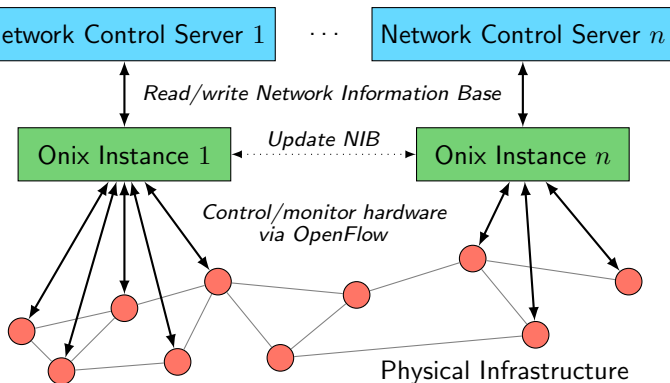
- Plattform zur Steuerung Software-definierter Netzwerke
 - Implementierung der Netzwerksteuerlogik als verteilte Anwendung
 - Plattform übernimmt Interaktion mit der Hardware
- Zentrale Datenstruktur: *Network Information Base (NIB)*
 - Repräsentation des aktuellen Netzwerkzustands
 - Verwaltung von Netzwerkelementen (z. B. Knoten, Verbindungen)
 - Zugriff aus Steueranwendungen
 - Aufruf von Methoden zum Lesen und Schreiben von Einträgen
 - Registrierung für Benachrichtigungen über Zustandsänderungen
 - Nach Änderungen am NIB erfolgt die Aktualisierung der entsprechenden physischen Netzwerkelemente in der Regel asynchron

■ Literatur



Teemu Koponen, Martin Casado, Natasha Gude, Jeremy Stribling et al.
Onix: A distributed control platform for large-scale production networks
Proceedings of the 9th Symposium on Operating Systems Design and Implementation (OSDI '10), S. 351–364, 2010.


■ Aufbau des Gesamtsystems



■ Mechanismen für verbesserte Skalierbarkeit

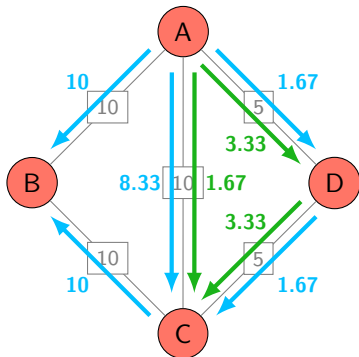
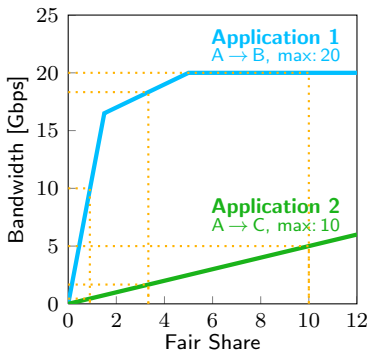
- Partitionierung des NIB und Aufteilung auf mehrere Onix-Instanzen
- Zusammenfassung von Netzwerkteilen zu *aggregierten Knoten*



- Nichtöffentliches WAN zur Verbindung der Google-Datenzentren
 - Übertragung von Nutzerdaten-Backups (z. B. E-Mails, Videos)
 - Abwicklung von Zugriffen auf verteilte Datenspeicher
 - Synchronisation von Anwendungszuständen
- Ziele
 - Zentrale Steuerung des Netzwerkverkehrs
 - Effizientere Auslastung der Netzwerkverbindungen
- Umsetzung
 - Implementierung auf Basis von (unter anderem) Onix und OpenFlow
 - Konstruktion eigener B4-Switches aus Standard-Hardware
- Literatur
 -  Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong et al.
B4: Experience with a globally-deployed software defined WAN
Proceedings of the 2013 SIGCOMM Conference, S. 3–14, 2013.



- Problem: Aufteilung der Übertragungskapazitäten auf Anwendungen
- Lösung: *Traffic Engineering*
 - Gewichtung von Anwendungen mittels *Bandwidth Functions*
 - Ressourcenzuteilung durch schrittweise Erhöhung der jeweiligen Anteile
 - Dynamische Einrichtung von Netzwerktunneln



[Hinweis: Verkürztes und vereinfachtes Beispiel; ausführlich in [Jain et al.].]

- Infrastructure as a Service
 - Bereitstellung von Rechenkapazität und Datenspeicher
 - Dynamische Skalierbarkeit in beide Richtungen
- Eucalyptus
 - Open-Source-Implementierung für private IaaS-Clouds
 - Schnittstellen: Orientierung an Amazon EC2
 - Hierarchischer Aufbau zur Unterstützung separater Cluster
- Software-definierte Netzwerke
 - OpenFlow
 - Grundidee: Auslagerung und Zentralisierung der Netzwerksteuerlogik
 - Einheitlicher externer Zugriff auf die Flow-Tabellen von Switches
 - Onix: Implementierung der Steuerlogik als verteilte Anwendung
 - B4
 - Weitverkehrsnetz zwischen Google-Datenzentren
 - Steuerung von Datenflüssen mittels (unter anderem) OpenFlow und Onix

