

Middleware – Cloud Computing

Cloud-Infrastrukturen

Wintersemester 2022/23

Tobias Distler

Friedrich-Alexander-Universität Erlangen-Nürnberg
Lehrstuhl Informatik 4 (Verteilte Systeme und Betriebssysteme)



Lehrstuhl für Verteilte Systeme
und Betriebssysteme



FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT

Cloud-Infrastrukturen

Motivation

Eucalyptus

Software-definierte Netzwerke

- **Bereitstellung von Ressourcen**
 - Virtuelle Maschinen (VMs) auf Systemebene
 - Zuverlässiger und hochverfügbarer Datenspeicher
- **Dynamische Skalierbarkeit** in beide Richtungen
 - Hinzufügen weiterer virtueller Maschinen bei Bedarfsspitzen
 - Herunterfahren von virtuellen Maschinen bei zu geringer Auslastung
- Überlegungen 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
- **Herausforderungen**
 - Wie lässt sich eine skalierbare Cloud-Infrastruktur realisieren?
 - Wie können Cloud-Datenzentren effizient miteinander kommunizieren?

Cloud-Infrastrukturen

Motivation

Eucalyptus

Software-definierte Netzwerke

■ Motivation

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

■ Eucalyptus

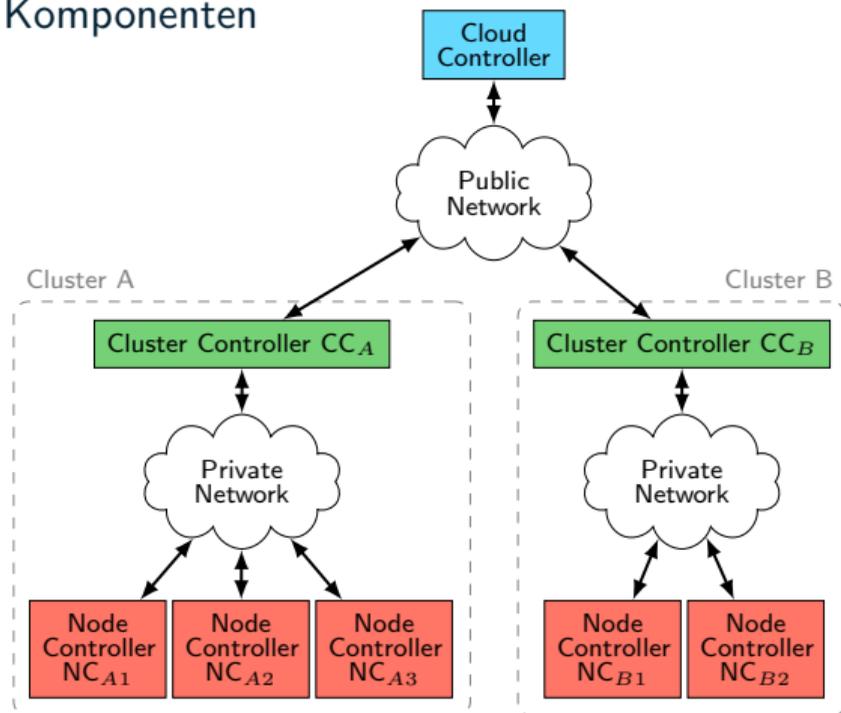
- Framework für **private bzw. hybride Infrastructure-as-a-Service-Clouds**
- Zielgruppe: Universitäten und kleinere Firmen
- Anlehnung an Amazon EC2 bzw. Amazon S3
 - Kommandozeilen-Tools zur Interaktion mit dem Framework
 - Client-Schnittstelle für Datenspeichersystem

■ Literatur



Daniel Nurmi, Rich Wolski, Chris Grzegorzczuk, Graziano Obertelli, Sunil Soman et al.
The Eucalyptus open-source cloud-computing system
Proc. 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 VM-Images
 - 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**
 - Bearbeitung von Anfragen des Cloud-Controller
 - Auswahl der Node-Controller für den Start virtueller Maschinen
 - Analyse der Kapazitäten für bestimmte VM-Typen
- 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: 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)
 - Verfügbarkeit im lokalen Image-Cache
 - Transfer aus dem Image-Archiv von Walrus
 - Anweisung an den Virtual Machine Monitor das VM-Image zu booten

→ Nutzer kann auf die virtuelle Maschine zugreifen (z. B. per `ssh`)

- 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**
 - Konfiguration und Überwachung durch Cluster-Controller
 - Realisierung der Isolation
 - Einrichtung eines separaten virtuellen Netzwerks (VLAN) für jeden Nutzer
 - Jedes virtuelle Netzwerk verwendet ein eigenes IP-Subnetz
 - Cluster-Controller
 - * Isolation durch Firewall-Regeln
 - * Falls erforderlich Routing zwischen IP-Subnetzen
 - Einfluss auf Erreichbarkeit
 - Verwendung privater IP-Adressen → VMs von außen nicht zugänglich
 - Bei Bedarf Adressumsetzung von öffentlichen auf private IP-Adressen

Cloud-Infrastrukturen

Motivation

Eucalyptus

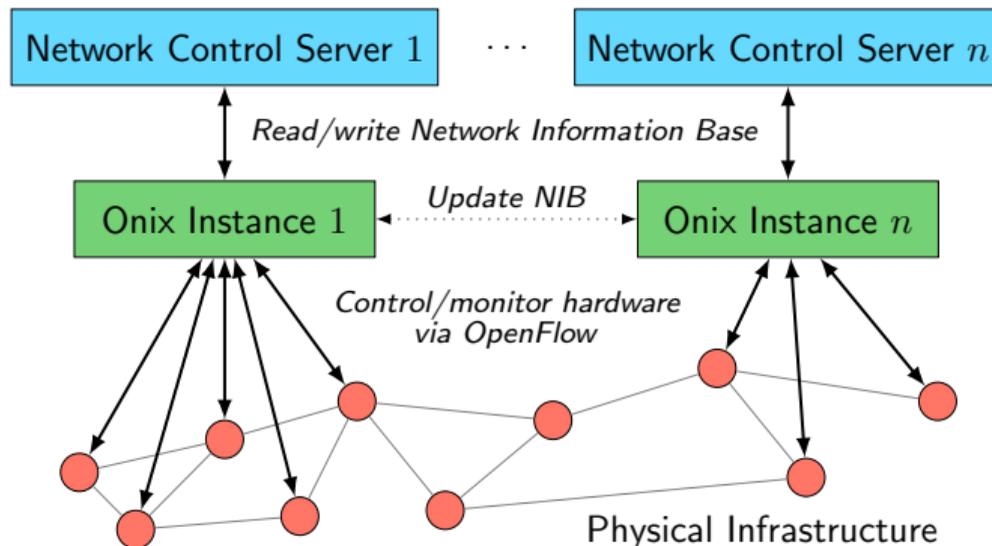
Software-definierte Netzwerke

- 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, Larry Peterson 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** (Beispiele)
 - Weiterleitung des Pakets an einen bestimmten Port
 - Verwerfen des Pakets
 - Kapselung und Weiterleitung des Pakets an den Controller
 - 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, Leon Poutievski et al.
Onix: A distributed control platform for large-scale production networks
Proc. 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**

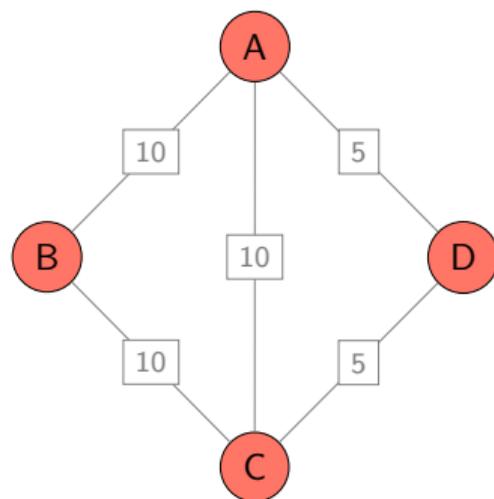
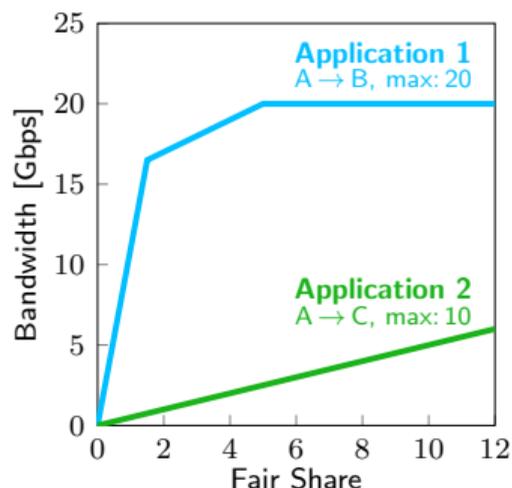
- **Nichtöffentliches Netzwerk** zur Verbindung der Google-Datenzentren
 - Übertragung von Nutzerdaten-Backups (z. B. E-Mails, Videos)
 - Abwicklung von Zugriffen auf verteilte Datenspeicher
 - Synchronisation von Anwendungszuständen
- Ziele
 - **Zentrale Steuerung des Netzwerkverkehrs**
 - Effizientere Auslastung der Netzwerkverbindungen
- Umsetzung
 - Implementierung auf Basis von (unter anderem) Onix und OpenFlow
 - Konstruktion eigener B4-Switches aus Standard-Hardware

- Literatur

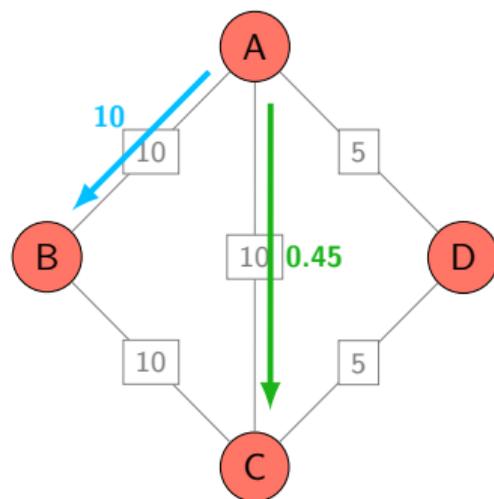
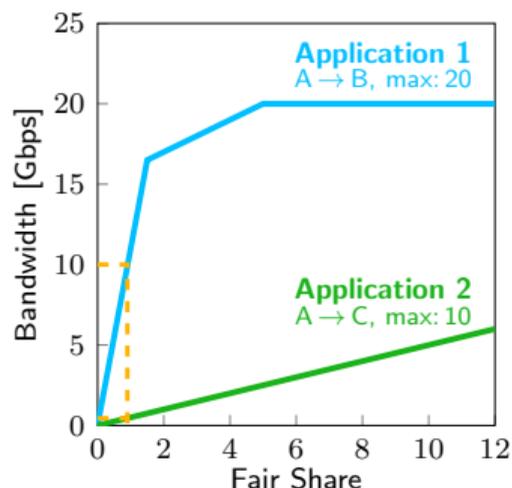


Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh et al.
B4: Experience with a globally-deployed software defined WAN
Proceedings of the 2013 SIGCOMM Conference, S. 3–14, 2013.

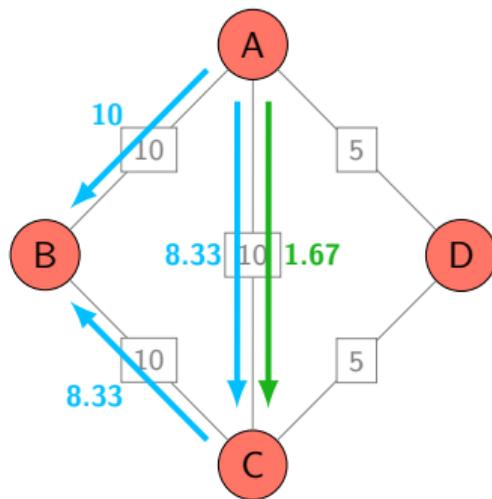
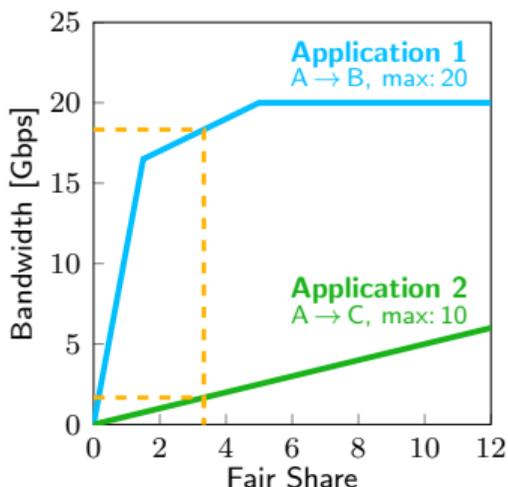
- Problem: Aufteilung der Übertragungskapazitäten auf Anwendungen
- Lösung: *Traffic Engineering*
 - Gewichtung von Anwendungen mittels **Bandwidth Functions**
 - Ressourcenzuteilung durch schrittweise Erhöhung der jeweiligen Anteile
 - Dynamische **Einrichtung von Netzwerktunneln**



- 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**



- 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**



- 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**

