technische universität
dortmund

Diplomarbeit

Dezentrale Scheduling-Verfahren für föderierte Cloud-Umgebungen

Marcel Krüger

04.10.2016

Gutachter:
Prof. Dr.-Ing. Olaf Spinczyk*
Dr. Lars Nagel, Universität Mainz

*Technische Universität Dortmund
Fakultät für Informatik
Lehrstuhl Informatik 12
Arbeitsgruppe Eingebettete Systemsoftware
http://ess.cs.tu-dortmund.de

In Kooperation mit:
Materna GmbH
Voßkuhle 37
44141 Dortmund

MATERNA
Information & Communications

Inhaltsverzeichnis

Abkürzungsverzeichnis

Capex	Capital Expenditure
CSP	Cloud-Service-Provider
DAG	Directed Acyclic Graph
FCS	FederatedCloudSim
IaaS	Infrastructure as a Service
IT	Informationstechnologie
ITK	Informations- und Telekommunikationstechnik
NIST	U.S. National Institute of Standards and Technology
Opex	Operational Expenditure
PaaS	Platform as a Service
QoS	Qualtity of Service
SaaS	Software as a Service
SLA	Service-Level-Agreement
SLO	Service-Level-Objective
VM	Virtuelle Maschine
XML	Extensible Markup Language
XSD	XML Schema Definition

Kapitel 1

Einleitung

Viele Technologien wurden entworfen um die Vision des *Utility-Computing* umzusetzen und Rechenleistung zur fünften Versorgungsleistungen nach Strom, Telefon, Wasser und Gas werden zu lassen (vgl. Buyya 2009 [9]). Als bisher erfolgreichstes Paradigma erweist sich *Cloud-Computing*. Der Einsatz von Technologien wie Virtualisierung und geeigneten Modellen zur Nutzung und Bereitstellung ermöglicht den variablen Bezug von Rechenressourcen als Service über ein Netzwerk.

1.1 Motivation

Die Vermietung seiner Ressourcen ist das wesentliche Geschäftsmodell eines Cloud-Service-Anbieters. Folglich ist es fundamental für ihn diese Ware erfolgreich zu vermarkten, ansonsten ist sein Unternehmen unrentabel. Scheduling-Verfahren bestimmen darüber, wie die Ressourcen genutzt werden, da sie die Allokation von Virtuellen Maschinen (VMs) auf Servern steuern. Sie haben entscheidenden Einfluss auf die Effektivität des Ressourcen-Managements eines Cloud-Service-Provider (CSP). Der Zusammenschluss mehrerer CSPs zu einer Föderation eröffnet neue Möglichkeiten Ressourcen vorteilhaft zu nutzen. Damit einher gehen allerdings auch neue Herausforderungen, denen sich die Scheduling-Verfahren stellen müssen. Insbesondere in dezentral organisierten Cloud-Umgebungen müssen sie dabei mit relativ geringen Informationen auskommen. Service-Level-Agreements (SLAs) haben sich als Grundlage der Geschäftsbeziehungen zwischen Providern und Kunden als auch zwischen Providern innerhalb einer Föderation etabliert. Das SLA-Management und das Finanz-Modell eines CSP ermöglichen nicht nur die ökonomische Bewertung der Dienstleistungen, sondern lassen auch Ansatzpunkte für Algorithmen erkennen um den Profit eines Providers zu steigern. Zur Untersuchung des Einflusses von Scheduling-Strategien auf die Wirtschaftlichkeit eines Cloud-Anbieters bieten sich Simulationen an, da Tests im laufenden Betrieb nicht praktikabel sind.

1.2 Zielsetzung der Arbeit

In dieser Arbeit wird das Simulations-Framework FederatedCloudSim um Strukturen erweitert, wodurch Scheduling von VMs in Cloud-Föderationen simuliert werden kann. Zudem wird ein Finanz-Modell entworfen und implementiert, das eine Abrechnung der Dienstleistungen eines Providers ermöglicht. Für das erweiterte Framework werden Scheduling-Verfahren entwickelt, welche insbesondere die Zielsetzung verfolgen die Wirtschaftlichkeit zu verbessern. Unter Berücksichtigung einer dezentralen Organisationsstruktur und von SLAs werden die Korrelationen zwischen den Strategien und der finanziellen Bilanz eines Providers und der Föderation anhand von reellen Cloud-Workloads untersucht. Es werden dabei Scheduling-Strategien analysiert, die statische oder dynamische Preisfindungs-Verfahren einsetzen und auf verschiedenen hierarchischen Ebenen Entscheidungen treffen.

1.3 Aufbau der Arbeit

Im folgenden Kapitel werden Begriffe definiert, Modelle und Konzepte vorgestellt und erklärt, die als Grundlage zum Verständnis dieser Arbeit dienen. In Kapitel 3 werden die Anforderungen an Scheduling-Verfahren erläutert, verschiedene Ausprägungen beschrieben und ihr Einsatz in Cloud-Föderationen erörtert. Dabei wird eine Abgrenzung dieser Arbeit zu anderen verwandten Abhandlungen vorgenommen. Zudem werden Modelle vorgestellt, welche im Rahmen dieser Abhandlung entwickelt wurden. Das Simualtions-Framework FederatedCloudSim wird in Kapitel 4 präsentiert und es wird aufgezeigt, anhand welcher Analyse-Ergebnisse, Entwürfe vorgenommen und Implementierungen durchgeführt wurden um seine Funktionalität zu erweitern. In Kapitel 5 werden verschiedene Scheduling-Verfahren auf realen Workloads ausgeführt, die Konfiguration dieser Simulationen beschrieben und ihre Erbebnisse ausgewertet. Eine Zusammenfassung der Erkenntnisse und einen Ausblick liefert abschließend Kapitel 6.

Kapitel 2

Grundlagen

In diesem Kapitel werden zu den wichtigsten Themengebieten dieser Arbeit die relevanten Grundlagen erklärt. Zunächst erfolgt eine generelle Einführung in das Cloud-Computing. Technische, funktionale und wirtschaftliche Aspekte werden beleuchtet, Klassifikationsmöglichkeiten aufgezeigt und davon ausgehend eine Abgrenzung zu anderen ähnlichen Konzepten begründet. Danach werden SLAs und ihr Nutzen für das Cloud-Computing bewertet, sowie der Aufbau von Cloud-Föderationen und die Besonderheiten des Scheduling in ihnen ausgeführt. Zum Abschluss werden elementare Kenntnisse zu Auktionen, und XML-basierten Textdateien vermittelt.

2.1 Cloud-Computing

Eine repräsentative Studie der Bitkom Research Gmbh zeigt, dass Cloud-Computing eine breite Marktdurchdringung erreicht hat und weiterhin ein kontinuierlicher Wachstumsmarkt in Deutschland ist (siehe BITKOM 2015 [7]). So setzen 54% aller Unternehmen in Deutschland Cloud-Computing ein und in weiteren 18% wird der zukünftige Einsatz diskutiert oder bereits geplant. Sogar 74% der Firmen der Informations- und Telekommunikationstechnik (ITK)-Branche nutzen es derzeit. Wie in Abbildung 2.1 dargestellt, wird für das Jahr 2018 ein Umsatzvolumen von 19,8 Millarden Euro prognostiziert.

2.1.1 Definition

Die enorme Beliebheit, der sich das Cloud-Computing in der Forschungsgemeinde erfreut, hat zu einer unübersichtlichen Anzahl an Definitionen geführt (siehe Geelan 2009 [16]). Insbesondere zwei werden hier erörtert, welche über die Zeit eine erhöhte Präsenz erlangt haben:

> I. Foster (2008) [15]: A large-scale distributed computing paradigm that is driven by economies of scale, in which a pool of abstracted, virtualized,

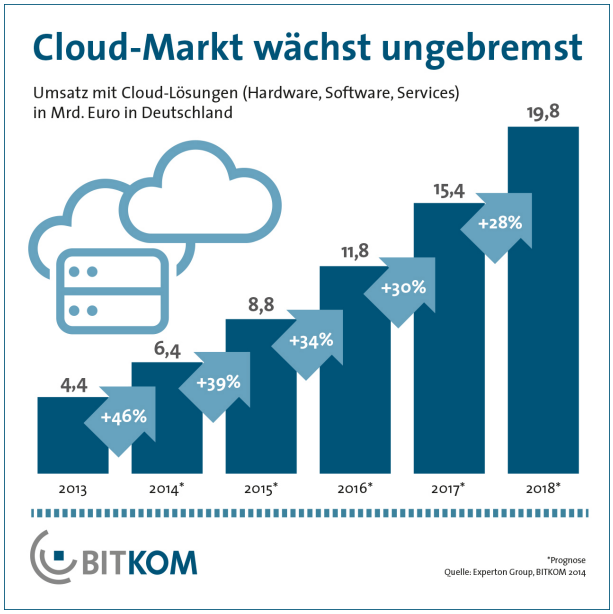

Abbildung 2.1: Wachstumsprognose des Cloud-Marktes (aus BITKOM 2014 [6])

> dynamically-scalable, managed computing power, storage, platforms, and services are delivered on demand to external customers over the Internet.
>
> NIST (2011) [32]: Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

Während die Definition von Foster vor allem die technischen Aspekte betont, bietet die Erklärung des U.S. National Institute of Standards and Technology (NIST) einen spezifischeren Blickwinkel an, der neben den essentiellen Charakteristiken auch das Konzept und Modelle für Nutzung und Bereitstellung umfasst (vgl. Teng 2011 [40]). Zusammenfassend lässt sich festhalten, dass Cloud-Computing für die Bereitstellung von Informationstechnologie (IT)-Ressourcen über ein Netzwerk steht. Dabei können Ressourcen sowohl IT-Infrastrukturen, wie auch Plattformen oder Anwendungen darstellen. Diese Dienstleistung wird als Service bezeichnet und muss bestimmte Eigenschaften aufweisen um dem Paradigma - im Sinne von Muster - des Cloud-Computing zugeordnet zu werden. Auf einige dieser bezeichnenden Merkmale und wie sich verschiedene Cloud-Computing-Konzepte anhand dieser Modelle klassifizieren lassen, wird nachfolgend detailliert eingegangen. Ein erstes Merkmal ist, dass beim Cloud-Computing technologische und wirtschaftliche Aspekte meist zusammen betrachtet werden (vgl. Baun et al. 2010 [3]). Weitere Merkmale lassen sich dem einen oder anderen Aspekt zuordnen.

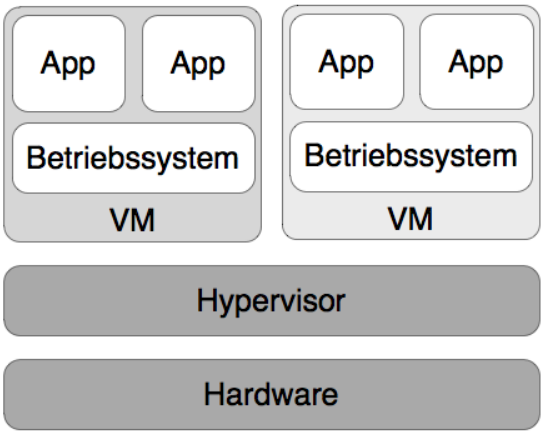

Abbildung 2.2: Virtualisierung

2.1.2 Informationstechnische Merkmale

Virtualisierung

Virtualisierung stellt die Basistechnologie des Cloud-Computing dar. Sie ermöglicht die Trennung logischer Systeme von der physischen Hardware. Software, insbesondere Betriebssystem und Daten, können so von der Infrastruktur abstrahiert werden. Es gibt verschiedene Virtualisierungskonzepte, die sich in ihrer Implementierung, Praxisrelevanz und Einsatzhäufigkeit unterscheiden. Wie in Abbildung 2.2 dargestellt, basiert Virtualisierung auf einer zusätzlichen Software-Schicht, Hypervisor genannt, deren Aufgabe es ist, die physikalischen Ressourcen zu verwalten und VMs zur Verfügung zu stellen (vgl. Cafaro 2011 [10]). Eine VM bildet die reale Infrastruktur nach und steht in Verbindung mit Cloud-Computing somit für eine Ressourcen-Einheit, durch die ein Service erbracht wird. Diese Abstraktion bietet eine Reihe von Vorteilen:

- Technische Komplexität wird verborgen.
- Die physischen Ressourcen können effizienter genutzt werden, indem sie auf mehrere, voneinander isolierte VMs aufgeteilt werden.
- Die virtuellen Instanzen können erzeugt, verworfen, dynamisch verändert und nahezu unterbrechungsfrei migriert werden.
- Die Verwaltung von Erzeugung und Konfiguration kann automatisiert werden.

Daraus resultierend ergeben sich Möglichkeiten zur Konsolidierung unterschiedlichster Anwendungsklassen auf wenigen physischen Komponenten, Effizienzsteigerung, Kostenersparnis, Erhöhung der Dienstgüte und passgenauen Konfiguration.

Mehrmandantenfähigkeit

Da einzelne virtuelle Instanzen sowohl gemeinsam auf einem System ausgeführt, als auch zwischen Systemen migriert werden können, sind Ressourcen in Pools zusammengefasst. Physische Infrastrukturen werden von mehreren Kunden gemeinsam genutzt.

Skalierbarkeit

Aufgrund der variablen Zuordnung der VMs zu einem Host und seinen Ressourcen können Kapazitäten dynamisch erhöht oder verringert werden, wodurch sie unbegrenzt erscheinen.

Messbarkeit

Die Ressourcennutzung kann automatisch kontrolliert, beobachtet und protokolliert werden, wodurch Transparenz für Dienstanbieter und Kunde entsteht.

Umfassender Netzwerkzugriff

Da die Services über ein Netzwerk bereitgestellt werden, wird ein Zugriff über beliebige Endgeräte ermöglicht.

2.1.3 Geschäftsbezogene Merkmale

Diensterbringung auf Anforderung/ Self-Service-Modell

Ohne Interaktion mit dem Anbieter können die Services vom Konsumenten auf Anforderung nutzbar sein.

Bedarfsgerechte Abrechnung

Aus der bedarfsorientierten Nutzung und entsprechenden Skalierung der Ressourcen, in Verbindung mit einem effektiven Monitoring, ergibt sich die Möglichkeit der verbrauchsgenauen Abrechnung, *Pay-per-Use* genannt.

IT-Ressourcen als Service

IT-Dienstleistungen anhand der beschriebenen Merkmale bedarfsgerecht, skalier- und konfigurierbar anzubieten, bündelt für Kunden und Anbieter mehrere Vorzüge. Kunden ermöglicht es flexiblen Ressourcenzugang mit geringer Kapitalbindung für Hardware, Software und Stellfläche, ohne Bewirtschaftungskosten und reduzierter Komplexität. Investitionskosten können deutlich reduziert und in operative Betriebskosten umgewandelt werden (vgl. Teng 2011 [40]). Anbietern ermöglicht es, ihre Investitionskosten aufgrund der Skaleneffekte

schnell zu amortisieren, sowie vorhandene IT-Infrastruktur wesentlich effizienter im Hinblick auf Auslastung, Energieverbrauch und resultierende Kosten zu betreiben. Wartung und Verwaltung kann vorwiegend automatisiert und vereinfacht werden.

2.1.4 Klassifikationen von Cloud-Umgebungen

Die Modelle zur Nutzung und Bereitstellung von Cloud-Computing-Services, welche auch Teil der Definition des NIST sind, haben sich als Standard zur Beschreibung und Klassifikation etabliert (siehe [32]).

Nutzungsmodelle

Cloud-Angebote können anhand ihrer organisatorischen Ausgestaltung unterschieden werden. Es wird zwischen *Private-, Community-, Public- und Hybrid-Clouds* unterschieden. Eine private Cloud wird nur von einer exklusiven Nutzergruppe, zum Beispiel einem Unternehmen oder einer Behörde intern genutzt, wobei sie durchaus von einem Fremdanbieter betrieben werden kann. Dagegen werden Public-Clouds zwar von einem Dienstleister betrieben, die angebotenen Services stehen aber der Öffentlichkeit zur Verfügung. In einer Community-Cloud kommen die Nutzer aus verschiedenen Unternehmen, welche die gleichen Ziele verfolgen. Hybride Clouds sind eine beliebige Mischform aus Private- und Public-Cloud.

Dienstleistungsmodelle

Der angebotene Service kann anhand seines Abstraktionsniveau einem Service-Modell zugeordnet werden (siehe Abbildung 2.3):

1. **Infrastructure as a Service (IaaS)** Über den Service erhält ein Nutzer Infrastruktur-Ressourcen wie Rechenleistung, Speicher, Netzwerkkomponenten, etc. und hat die Kontrolle über Betriebssyteme und Applikationen, die auf dieser Infrastruktur ausgeführt werden.

2. **Platform as a Service (PaaS)** Wird ein Service auf dieser Ebene bezogen, so erhält ein Nutzer eine Plattform, auf der zumeist die Entwicklung und Ausführung eigener Software unterstützt wird. Der Zugriff auf die zugrundeliegende Betriebssystem-Ebene entfällt. PaaS ist damit vor allem für Softwareentwickler interessant, die den Aufwand zur Erstellung einer Programmierumgebung vermeiden wollen.

3. **Software as a Service (SaaS)** Über ein Web-Interface, wie beispielsweise einen Browser, erhalten Kunden von Services dieser Ebene Zugang zu Anwendungssoftware. Darunterliegende Schichten sind vollständig abstrahiert. SaaS entspricht damit der Miete einer Applikation, die nach der tatsächlichen Nutzungszeit abgerechnet werden kann.

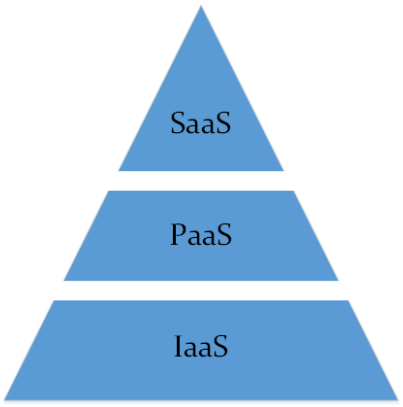

Abbildung 2.3: XaaS-Service-Modelle (Quelle: [43])

2.1.5 Vergleich mit verwandten Technologien

Cloud-Computing ist keine langfristig und stringent konzeptionierte Technologie. Vielmehr ist es eine Evolution aus einer Vielzahl von Vorläufertechnologien, Geschäftsmodellen und Ideen. Demzufolge gibt es vielfache Überschneidungen und Wechselwirkungen, die häufig auch zu Verwirrungen über das Konzept führen (vgl. Bedner 2013 [4]). Der Onlinehändler Amazon stellte seine für das Weihnachtsgeschäft notwendigerweise vorgehaltenen Ressourcen, die den Rest des Jahres brachlagen, erstmals im Jahr 2006 für die Allgemeinheit zur Verfügung, für viele die Geburtsstunde des Cloud-Computing (vgl. Reti et al. 2009 [29]). Der Begriff und einige konzeptionelle Teile sind allerdings schon älter. Zudem entwickelte sich Cloud-Computing stetig weiter und eine klar umrissene Abfassung und Abgrenzung kristallisierte sich erst im Laufe der Zeit heraus. Überschneidungen weisen insbesondere folgende Paradigmen auf:

- Als Zusammenschluss unabhängiger Computer, die sich als ein System präsentieren, entspricht Cloud-Computing der Definition eines **verteilten Systems** nach Tanenbaum (2003) [39].

- **Cluster-Computing** beschreibt die Vernetzung mehrerer Rechner zur Erhöhung der Rechenleistung und Verfügbarkeit, ohne dass das Modell Bereitstellung oder Nutzung spezifiziert und stellt damit eine Vorläufertechnologie zum Cloud-Computing dar.

- Beim **Grid-Computing** werden Computercluster, die an einen Standort gebunden sind, zu virtuellen Supercomputern zusammengeführt. Grid- und Cloud-Computing weisen eine Vielzahl von Gemeinsamkeiten auf und es lassen sich Entwicklungen und Forschungen des einen häufig für den anderen Bereich adaptieren. Die Konzepte lassen sich allerdings aufgrund einiger Merkmale sinnvoll abgrenzen. Grid-Computing beinhaltet keine ökonomischen Aspekte, sondern zielt allein auf die optimale Auslastung vorhandener Ressourcen zur Lösung mathematischer und wissenschaftlicher

Probleme. Zudem fehlt die Flexibilität des Cloud-Computing. Ein Zusammenschluss wird meist langfristig geplant. Insbesondere sind Grids aber hierarchisch strukturiert und besitzen eine zentrale Steuerungs-Instanz (vgl. Foster 2008 [15]).

- **Utility-Computing** ist eine Idee, die in den späten 60er Jahren populär gemacht wurde. Rechenleistung sollte den Versorgungsleistungen wie Strom, Telefon, Wasser und Gas gleich, bereitgestellt werden. Damit beinhaltet diese Vision viele wesentliche Kriterien des Cloud-Computings und es ist durchaus berechtigt zu sagen, dass Cloud-Computing eine Form von Utility-Computing ist. Unterscheiden lassen sich die Konzepte nur in einzelnen Merkmalen, so werden o.g. Leistungen dauerhaft bereitgestellt, während Cloud-Computing für flexible Anbieterauswahl und kurze Vertragslaufzeiten steht.

2.2 Service Level Agreements

SLAs bilden die formale Grundlage, um funktionale und nicht-funktionale Konditionen der Erbringung eines Services zu spezifizieren (vgl. Wieder et al. 2011 [42]). Das Ziel ist es die Qualität der Leistung zu beschreiben und festzulegen. Dabei erfolgt eine Beschreibung des Service, indem Nutzen und Funktionalität meist aus Sicht des Service-Kunden dargestellt sind. Zudem sind konkrete Ziele anhand messbarer und realistischer Kennzahlen vereinbart, Service-Level-Objectives (SLOs) genannt. Die Verantwortlichkeiten der Vertragspartner sind geregelt, die Lieferung des beschriebenen Services und die Einhaltung der Qualitätskennzahlen obliegt dabei dem Service-Provider. Das Service-Level-Management gliedert sich üblicherweise in zwei Phasen, die Vereinbarung der Leistungen und ihre Überwachung, SLA-Monitoring genannt (vgl. Bedner 2013 [4]). Individuelle Übereinkünfte sind eher unüblich, da sie eine automatische Aushandlung und Auswertung erschweren. Um eben diese zu ermöglichen, ist eine maschinenlesbare Form nötig. Eine Darstellung der Struktur in Extensible Markup Language (XML) ermöglicht dies. Beispiele für SLOs, insbesondere für diese Arbeit relevante, sind:

- Angaben einer prozentualen Verfügbarkeit des Service.
- Preise und Kosten, insbesondere Strafzahlungen im Falle eines SLA-Bruchs.
- Migrationsbeschränkungen, die beispielsweise ein Auslagerungsverbot zu einem Rechenzentrum in einem anderen Land beinhalten.
- spezifische Angaben zu VMs, wie Anzahl der CPU-Kerne, deren Geschwindigkeit und Größe des Hauptspeichers.

Wird eine der Vereinbarungen nicht eingehalten, SLA-Bruch genannt, so wird eine Vertagsstrafe fällig. Eine Kapselung verschiedener Kriterien wird durch die Definition von

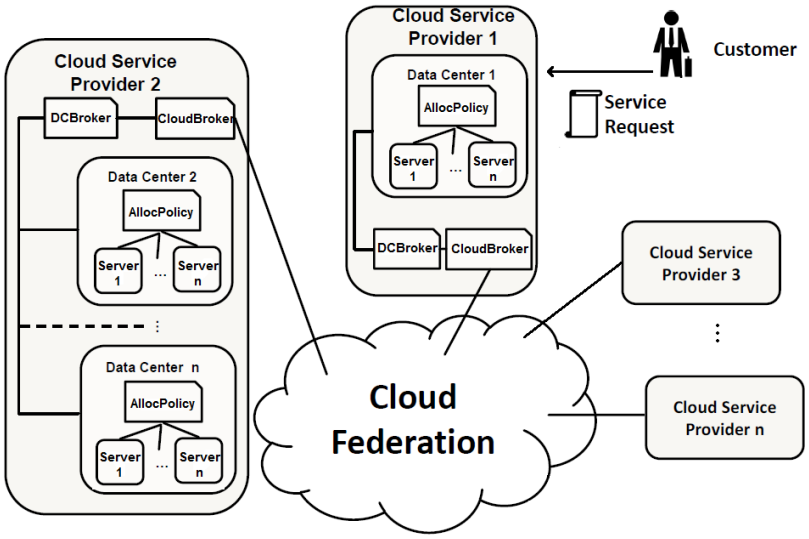

Abbildung 2.4: Aufbau einer Cloud-Föderation (Quelle: [28])

Dienstgüten ermöglicht, die sodann mit Bronze-, Silber-, Gold- oder Platin-Level bezeichnet werden.

2.3 Cloud-Föderationen

Wie anhand der Merkmale des Cloud-Computing beschrieben, ist die Vermietung vorhandener Ressourcen das Kernelement des Geschäftsmodells von Anbietern eines Cloud-Service, im Folgenden CSP genannt. Die effiziente Nutzung seiner Ressourcen ist damit geschäftskritisch. Der entscheidende Einflussfaktor für die Erreichung dieses Ziels ist, die Verteilung der erzeugten VMs auf die zur Verfügung stehenden Hosts. Durch geschickte Strategien lässt sich dadurch die Auslastung der Rechenzentren eines CSP optimieren. Da die Nutzer eines Service, die angefragten Ressourcen meist nicht, oder nicht dauerhaft nutzen, ergibt sich für Anbieter die Möglichkeit mehr Ressourcen zu vermieten als tatsächlich vorhanden sind. Diese Überprovisionierung birgt allerdings die Gefahr, dass die vertraglich zugesagten Leistungen nicht erfüllt werden können, falls es zu Lastspitzen kommt. VMs durch Live-Migration in angeschlossene Hosts oder sogar andere Rechenzentren zu verschieben, ermöglicht es diese Lastspitzen abzufangen. Durch die Migration entstehen geringe Ausfallzeiten. CSPs stehen vor der Herausforderung abzuwägen, zwischen dem Risiko geschlossene SLAs durch zu viele Ausfallzeiten zu brechen oder eine möglichst hohe und dadurch gewinnbringende Auslastung zu erreichen. Ein Lösungsansatz für dieses Problem stellen *Cloud-Föderationen* dar. Dies ist der Zusammenschluss mehrerer CSPs zu einer Kooperationsgemeinschaft (vgl. Kohne et al. 2014 [27]). Dieses Konzept erweitert

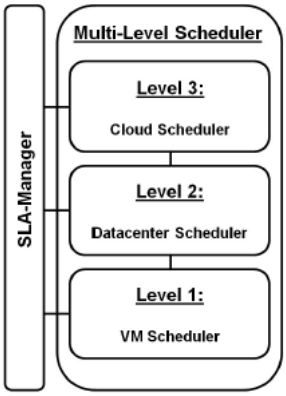

Abbildung 2.5: Anbindung des SLA Managers an den Multi-Level-Scheduler nach Kohne et al. (2013) [26]

Cloud-Computing-Infrastrukturen im Sinne des zugrundeliegenden Geschäftsmodells um eine weitere Ebene. Ein Anbieter wird seinerseits zum Kunden und bezieht IT-Ressourcen über ein Netzwerk. Es ergeben sich neue Möglichkeiten, auf Variationen bei Ressourcen-Anfragen zu reagieren. Neue Anfragen müssen nicht abgelehnt werden, was neben den entgangenen Einnahmen meist auch einen Verlust von Reputation und zukünftigen Kunden bedeutet hätte. Bei Lastspitzen können Ressourcen aus der Föderation angemietet werden und andersherum können ungenutzte Kapazitäten Kooperationspartnern angeboten werden. Beides kann dazu beitragen, den Profit eines Providers zu erhöhen (vgl. Goiri et al. 2010 [17]). Für kleinere CSPs bietet sich zudem die Gelegenheit, ihr Portfolio an Diensten zu erweitern, indem Services von Föderationspartnern in Gänze übernommen werden.

Es gibt zwei Ansätze zur Strukturierung einer Föderation. Bei einem zentralen Ansatz existiert eine Verwaltungseinheit, an der alle Mitglieder angeschlossen sind. Diese gibt, auf Basis der Informationen über alle CSPs, Empfehlungen oder entscheidet selbstständig über das Scheduling von VMs. Demgegenüber ist jeder CSP in einem dezentral organisierten Verbund allein selbstverantwortlich für seine Migrationsentscheidungen. In Abbildung 2.4 ist das abstrakte Modell einer Föderation dargestellt, wie es von Kohne et al. (2013) [26] beschrieben ist. Neben dem Kunden, der Service-Anfragen stellen kann, sind CSPs mit ihrer Struktur gezeigt. Jeder Anbieter kann ein oder mehrere Rechenzentren betreiben, die jeweils über eine Anzahl von Servern verfügen. Um nun Kundenanfragen weiterleiten und für bestehende Services erzeugte VMs zur Lastverteilung migrieren zu können, besitzt jeder Provider Scheduling-Komponenten auf drei verschiedenen Ebenen. Der *Cloud-Broker* eines CSP ist für die Kommunikation zwischen den Providern verantwortlich. Für die Steuerung der Abläufe zwischen mehreren Rechenzentren eines CSPs ist ein *Data-Center-Broker* zuständig. Innerhalb eines Rechenzentrums schließlich regelt eine *Allocation-Policy* die Vorgänge. Abbildung 2.5 verdeutlicht nochmals diese drei Level. Zu

sehen ist zudem die Anbindung eines SLA-Managers. SLAs werden nach diesem Modell nicht nur zwischen einem Anbieter und Kunden ausgehandelt, sondern bilden auch die Grundlage der Geschäftsbeziehung zwischen CSPs. Der Manager ist für die Aushandlung und Überwachung der Vereinbarungen zuständig. Neben diesem abstrakten, strukturellen Modell von Cloud-Föderationen stellen KOHNE et al. (2013) zudem ein Ablaufmodell vor, das den allgemeinen Ablauf zur Behandlung von Service-Anfragen unter Beachtung der SLAs beschreibt (siehe Abbildung 2.6 aus [26]). Es obliegt dem SLA-Manager die Gefährdung der Vereinbarungen zu überwachen und gegebenenfalls die erste Scheduling-Instanz zu informieren, dass eine Migration zur Auflösung der SLA-Gefährdung nötig ist. Kann ein Service nicht erbracht werden, so ist eine Migrationsanfrage an die jeweils nächst höhere Ebene eine Alternative. Führt dieses Vorgehen zu keiner Lösung, so muss eine Umverteilung der VMs auf ihrem lokalen Rechenzentrum angestoßen werden. Dies führt entweder zu einer unbedrohlichen Lastverteilung oder führt zu einer veränderten VM-Auswahl, für die Migrationsanfragen gestellt werden können. Das Modell zeigt zudem die Schnittstellen auf, an denen durch strategische Entscheidungen Einfluss auf die Migration von VMs genommen wird.

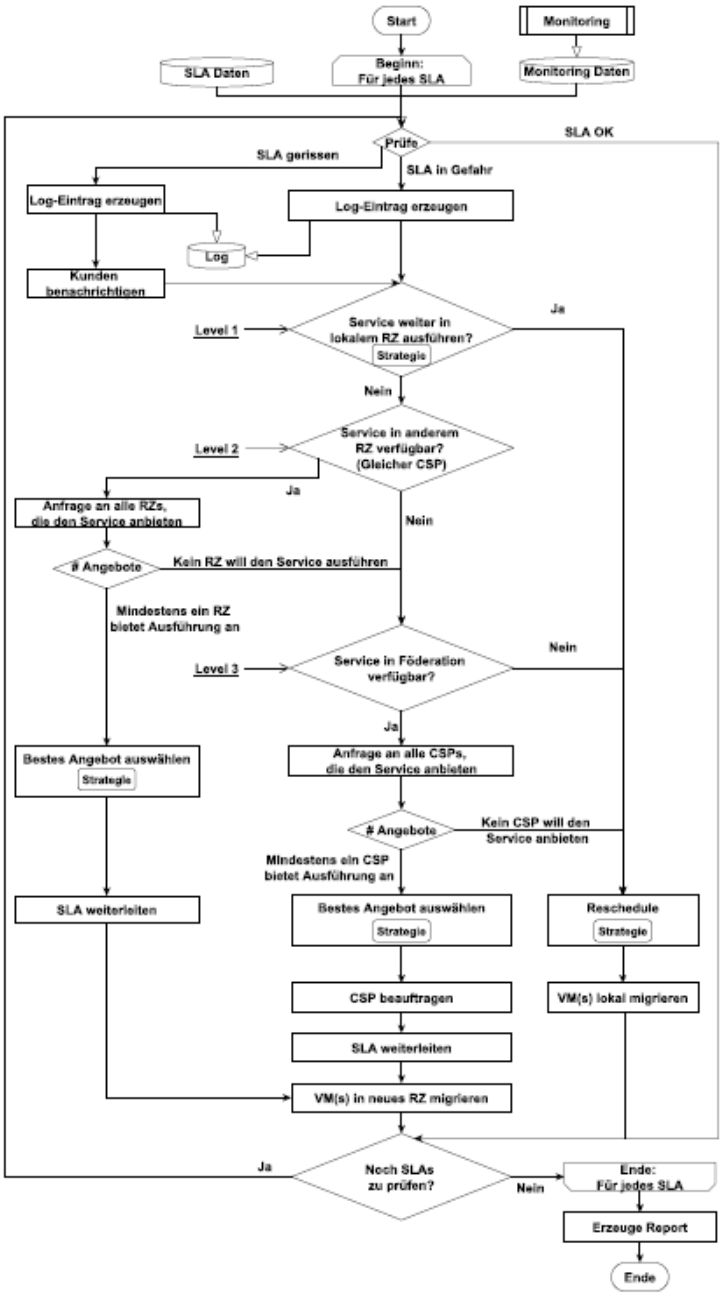

Abbildung 2.6: Grafische Darstellung des SLA-basierten Multi-Level-Schedulings nach Kohne et al. (2013) [26]

Kapitel 3

Scheduling-Verfahren

3.1 Einführung

Der wörtlichen Übersetzung nach bedeutet Scheduling die Planung eines Ablaufs. Dabei ist das Grundproblem der Scheduling-Theorie zu planen, wie mit einer begrenzten Anzahl an Ressourcen bzw. Maschinen eine Menge von Aufträgen möglichst günstig zu bearbeiten ist (vgl. Pinedo 2015 [35]). Was genau günstig bedeutet, muss für jedes Problem explizit spezifiziert werden. Ein *Schedule* ist folglich ein genauer Ablaufplan, in welchem die Zuweisung von Aufträgen zu Ressourcen erfolgt. Dabei werden die in einem Scheduling-Problem spezifizierten Vorgaben eingehalten. Graham et al. (1979) [20] haben ein Schema definiert, das genutzt wird um Scheduling-Probleme zu formalisieren und aufgrund ihrer entscheidenden Merkmale zu klassifizieren. Dieses Schema besteht aus drei Feldern

$$\alpha|\beta|\gamma \quad .$$

In diese Felder wird zur Klassifikation eines Scheduling-Problems jeweils ein eindeutiger Bezeichner für jeweils ein Merkmal eingetragen. Die Merkmale beziehen sich dabei entweder auf die Maschinenumgebung (α), die Aufgaben-Eigenschaften (β) oder die untersuchten Zielfunktionen (γ). Es werden drei Grundtypen von Maschinenumgebungen unterschieden, die wiederum in unterschiedliche Ausprägungen unterteilt sind (vgl. Gondek 2011 [19]):

1-Maschinen-Probleme:
Es ist der Ablauf auf einer einzelnen Maschine zu planen.

Parallelmaschinen-Probleme:
Es wird unterschieden zwischen Problemen, bei denen identische Maschinen parallel arbeiten, solchen, bei denen die parallelen Maschinen unterschiedliche Geschwindigkeiten haben und solchen, bei denen die unterschiedlichen Geschwindigkeiten zudem von dem zu bearbeitenden Auftrag abhängen.

Shop-Scheduling-Probleme:

In dieser Kategorie werden komplexere Strukturen von Umgebungen zusammengefasst. In einem *Flow-Shop* gibt es verschiedene Maschinen und alle werden von allen Aufträgen in derselben Reihenfolge belegt. Dagegen besitzt in einem *Job-Shop* jeder Auftrag eine individuelle Abfolge für die Gesamtmenge der Maschinen. In einem *Open-Shop* müssen wiederum alle Maschinen durchlaufen werden, allerdings ist ihre Reihenfolge für die Aufträge beliebig.

Es können durch Kombination von Parallel-Maschinen-Problemen mit Shop-Scheduling-Problemen noch komplexere Strukturen spezifiziert werden. Durch spezielle Eigenschaften einer Aufgabe (β) können zusätzliche Restriktionen beschrieben werden, welche durch das Schedule eingehalten werden müssen. Dabei sind verschiedene Bedingungen denkbar, wie zum Beispiel:

- Die Aufträge haben fixe Startzeitpunkte.
- Die Aufträge dürfen nicht in der Bearbeitung unterbrochen werden.
- Es ist eine festgelegte Reihenfolge von bestimmten Aufträgen einzuhalten, die meist durch einen gerichteten Graphen beschrieben ist.
- Aufträge blockieren eine Ressource auch nach ihrer Abarbeitung, falls die nächste geplante Ressource dieses Auftrages nicht frei ist.

Schließlich werden im γ-Feld Kriterien aufgeführt. Durch diese wird die Zielsetzung des Schedulings beschrieben, nach der es optimiert werden soll. Viele Beispiele für mögliche Zielfunktionen und eine umfassende Beschreibung der gängigen Bezeichnung der Problem-Parameter finden sich in [35] und [19].
Die Verfahren zur Findung einer Lösung für diese Probleme werden Scheduling-Algorithmen genannt und stellen eine Abfolge von Anweisungen dar, aufgrund derer eine Lösung des Problems in Form einer Zuordnung erzeugt wird. Für Scheduling-Algorithmen, welche in Bezug auf mehrere, sich widersprechende Zielsetzungen hin formuliert sind, kann kein effizienter Scheduling-Algorithmus existieren. In diesem Fall wird nach einem Kompromiss zwischen den zuwiderlaufenden Zwecken oder einer Kombination von Verfahren gesucht (vgl. Teng 2011 [40]).

3.2 Scheduling in Cloud-Umgebungen

In Cloud-Umgebungen wird zwischen Scheduling auf Nutzer-Ebene und System-Ebene unterschieden. Das Scheduling auf ersterer Ebene sorgt für eine Zuordnung von Konsumenten zu bereitgestellten Services und es stehen wirtschaftliche Anliegen wie das Gleichgewicht von Angebot und Nachfrage oder Minimierung von Kosten bei variierenden Nutzungsanfragen im Fokus der Optimierung. Auf System-Ebene bezieht sich Scheduling auf

die Zuweisung von Kundenaufträgen zu den verfügbaren Server-Ressourcen, beispielsweise Prozessorzeit und Hauptspeicher (vgl. [40]). Betrachtungsgegenstand dieser Ausarbeitung sind die Gestaltung und Auswirkungen des Schedulings auf System-Ebene, insbesondere aus Provider-Sicht. Weitere Erläuterungen zum Scheduling beziehen sich darauf. Durch ein Virtualisierungskonzept werden die physikalischen Ressourcen in Form von VMs bereitgestellt. Aufträge werden einzelnen VMs zugeordnet, sodass diese für die Erbringung des Service verantwortlich sind. Ein Scheduling plant die Belegung der verschiedenen Ressourcen durch die VMs unter der Beachtung von Bedingungen. Diese Bedingungen können beispielsweise Zeitbeschränkungen sein, bis zu welcher Fertigstellungsfrist ein Auftrag ausgeführt sein muss, Budgetbeschränkungen, so dass nur ein bestimmtes Betrag investiert werden darf oder Ressourcen können in ihrem Umfang eingeschränkt sein. Es gibt zumeist Überschneidungen zu den Optimierungskriterien. Eine Fertigstellungsfrist wäre ein Beispiel für eine Bedingung, während die Vorgabe einen Auftrag in minimal notwendiger Zeit zu bearbeiten eine Vorgabe zur Optimierung wäre. Solche Kriterien werden zur in der Optimierung genutzt und geben das Ziel des Scheduling-Prozesses vor. Die Ressourcen-Nutzung soll erhöht, die Anzahl der erfolgreich abgeschlossenen Aufträge vergrößert und Antwortzeiten minimiert werden. Es lässt sich eine Optimierungsfunktion entwerfen, welche die Bedingungen enthält und es wird versucht, diese Funktion in Bezug auf die Zielkriterien zu optimieren. Über den Wert einer Optimierungsfunktion kann die Qualität einer Lösung beurteilt und damit vergleichbar gemacht werden.
Es sind zwei Akteure in Cloud-Computing-Szenarien zu unterscheiden. Die Nutzer eines Cloud-Service haben Interesse daran, dass ihr Auftrag ausgeführt wird und haben den Anspruch, dass er zuverlässig, schnell und günstig bearbeitet wird. Im Gegensatz dazu verfolgen Service-Anbieter, die CSPs, unternehmerische Interessen. Zur Erhaltung und Steigerung des Unternehmenswertes, der ein zentrale Aufgabe der strategischen Geschäftsführung eines Unternehmens ist, zielen Unternehmen auf eine Steigerung ihres Profits durch Maximierung des *Return on Investments*. Das bedeutet, die Rendite auf das eingesetzte Kapital soll optimiert werden (vgl. Copeland et al. 2002 [13]). Folglich sind Provider darum bemüht, dass ihre Ressourcen nicht ungenutzt bleiben und damit aus wirtschaftlicher Sicht betrachtet ohne Einnahmen aber mit laufenden Betriebskosten verfallen.
Das Problem der Zuweisung von Aufträgen zu verteilten Ressourcen gehört zu der Klasse bekannter NP-schwieriger Probleme (siehe Kafil und Ahmad 1998 [22]). Für solche Probleme existieren keine Algorithmen, welche eine optimale Lösung in Polynomialzeit berechnen. Eine vollständige Suche ist zudem bei einer vorliegenden großen Anzahl von Lösungsmöglichkeiten nicht praktikabel [1]. Daher werden Heuristiken eingesetzt um in angemessener Zeit gute Lösungen zu finden. Kalra und Singh (2015) [23] bieten hierzu einen umfassenden Überblick.

[1] Eine Auflistung bekannter Scheduling-Probleme und ihre Komplexität bietet `http://www.mathematik.uni-osnabrueck.de/research/OR/class/`

Scheduling beinhaltet in diesem Szenario zwei Aufgaben - zum einen eine initiale Verteilung der VMs zu Hosts und zum anderen die Neuzuweisung im Falle eines Re-Schedulings. Falls eine VM nicht weiter auf dem ihr zugewiesenen Host ausgeführt wird, aufgrund von Fehlern, mangelnden Ressourcen oder strategischen Entscheidungen, so wird diese und unter Umständen weitere, in einem Re-Scheduling neu zugewiesen (vgl. KOHNE et al 2014 [27]). Die Migration von VMs lässt sich in zwei Phasen unterteilen. Zunächst wird eine Auswahl von möglichen Kandidaten getroffen. Im Anschluss folgt eine Selektionsphase der Hosts, um das Migrationsziel eines Kandidaten zu bestimmen (vgl. BELOGLAZOV [5]). In beiden Phasen können Algorithmen zur Optimierung eingesetzt werden.

3.3 Klassifikation von Scheduling-Verfahren

Scheduling-Verfahren können anhand folgender Kriterien differenziert werden:

Statisches (offline) versus dynamisches (online) Scheduling

Beim statischen Scheduling ist detailiertes Wissen über alle zu bearbeitenden Aufträge und alle Ressourcen im Vorhinein verfügbar, so dass ein Schedule vorab für die gesamte Laufzeit erstellt werden kann. Demgegenüber steht das dynamische Scheduling. Aufträge müssen zur Laufzeit auf Anforderung zugewiesen werden, mit eingeschränktem Wissen, beispielsweise über ihre Laufzeit oder ihren Ressourcenbedarf.

Zentrales, dezentrales oder hierarchisches Scheduling

Zentrales Scheduling zeichnet sich durch die Existenz eines Scheduler aus, welcher alle Informationen und Kontrolle über die Aufträge und verfügbaren Ressourcen hat. Beim dezentralen Scheduling hingegen fehlt diese Meta-Instanz. Stattdessen gibt es mehrere verteilte Scheduler, die untereinander kommunizieren und Informationen austauschen können. Für ein Schedule müssen sie überein kommen, aber jeder für sich trifft unabhängige Scheduling-Entscheidungen, basierend auf seinen lokalen Informationen. Das hierarchische Scheduling zeichnet sich aus durch Scheduler auf verschiedenen Ebenen. Die unterste Ebene ist dabei immer die Zuteilung der Ressourcen. Darauf aufbauend können ein oder mehrere Ebenen sein, eine Scheduling-Entscheidung wird auf mehrere Teilhandlungen aufgebrochen. Es können bei dieser Klassifikation Mischformen aller drei Kriterien auftreten, beispielsweise zwei Hierarchieebenen mit jeweils einem zentralen Scheduler.

Präemptives versus nicht-präemptives Scheduling

Während Aufträge innerhalb eines präemptiven Schedulings unterbrochen, gegebenenfalls migriert und fortgesetzt werden können, sind Ressourcen beim nicht-präemptiven Scheduling bis zur Fertigstellung eines Auftrags gebunden.

Direkte Verarbeitung versus Stapelverarbeitung

Wenn ein Auftrag im Moment seines Eintreffens einer VM und einem Host zugewiesen wird, entspricht dies einer direkten Verarbeitung. Wird hingegen zu diskreten Zeitpunkten jeweils eine Menge von VMs oder Aufträgen, welche innerhalb des vergangenen Zeitintervalls zugewiesen wurden, zusammen verarbeitet, so wird dies als Stapelverarbeitung bezeichnet (*englisch: Batch-Mode-Scheduling*).

Unabhängiges versus Workflow-Scheduling

Diese Differenzierung beruht auf der Verknüpfung der Aufträge. Bestehen keine Abhängigkeiten, so wird von unabhängigem Scheduling gesprochen. Dem gegenüber steht das Workflow-Scheduling, welches sich durch zeitliche Zusammenhänge der Aufträge auszeichnet. Wenn diese oder Teilmengen von ihnen in einer bestimmten Reihenfolge ausgeführt werden müssen, so können diese Abhängigkeiten in einem gerichteten azyklischen Graphen (DAG) dargestellt werden. Ein Auftrag kann erst dann einer Ressource zugewiesen werden, wenn seine Vorgänger in diesem DAG beendet wurden.

Eine weitere Möglichkeit zur Klassifikation von Scheduling-Verfahren bietet die Betrachtung der Zielfunktion eines Schedules. Es lassen sich anhand der Sichtweise der unterschiedlichen Akteure ihre Ziele in applikationsbezogene und ressourcenbezogene Bestrebungen einteilen (siehe Abbildung 3.1). Während für einen Nutzer zumeist die Bearbeitungszeit und die Gebühr eines Service relevant sind, liegt es im vordergründigen Interesse eines Providers die Ressourcen bestmöglich auszunutzen und größtmöglichen Profit zu erwirtschaften. Mittelbar ermöglicht ihm , seine Investition in eine Infrastruktur zu schützen und Risiken der Unter- bzw. Überlastung zu mitigieren. Ein Scheduling-Verfahren kann nun die Funktion haben, einen dieser Parameter zu optimieren oder auch eine bestmögliche Kombination mehrerer Kriterien anzustreben.

3.4 Dezentrale Verfahren für Cloud-Föderationen

In Abschnitt 2.3 wurden Cloud-Föderationen vorgestellt und zwei Strukturierungsansätze gegenüber gestellt. Eine zentral organisierte Struktur bietet dabei die Vorteile, dass aufgrund der Zusammenführung von Informationen effiziente Scheduling-Entscheidungen getroffen werden können. Zudem können viele Forschungsergebnisse aus dem technisch verwandten Grid-Computing übernommen werden (siehe [15]). Ein verteilter Ansatz hingegen erscheint realistischer, denn nur auf diese Weise sind die einzelnen Provider in der Lage ihrer eigenen Strategie, ihren Bedürfnissen und Beschränkungen entsprechend Entscheidungen zu treffen. Zudem sind zentrale Strukturen nicht zuletzt deswegen selten in kommerziellen Clouds vertreten, weil die Provider Informationen über ihre Auslastung und Infrastruktur nicht preisgeben wollen (vgl. Kohne et al. [27]). Die weiteren Untersuchungen dieser Ar-

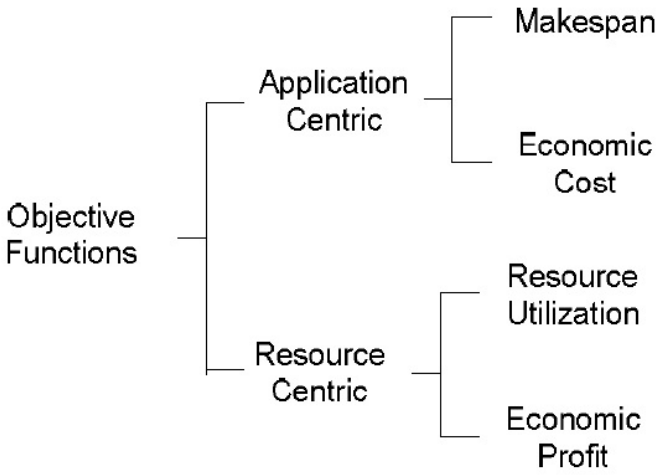

Abbildung 3.1: Systematik der Zielfunktion (vgl. DONG UND AKL 2006, Abbildung 3 [14])

beit beschränken sich daher auf dezentral organisierte Cloud-Föderationen.
Eine dezentrale Struktur einer Föderation führt dazu, dass die unterschiedlichen Bedürfnisse der Teilnehmer an ein Scheduling mitunter gegensätzlich sind. Die jeweiligen Anforderungen beruhen auf lokalen Gegebenheiten. Die Parteien kommunizieren miteinander und tauschen dabei Entscheidungen und gegebenenfalls Teile ihrer privaten Informationen aus. Dies ist erforderlich, da diese Informationen zur Bewertung von Scheduling-Alternativen notwendig sind. Die generellen Herausforderungen eines dezentralen Scheduling sind daher nach WELLMAN et al. (2001) [41]:

1. Welcher Struktur unterliegt die Kommunikation zwischen den Teilnehmern einer Föderation?

2. Wie wird eine Übereinkunft erzielt und die finale Entscheidung über ein Scheduling getroffen?

3. Wie werden die relevanten, privaten Informationen repräsentiert? Und wodurch sind die Teilnehmer motiviert diese preiszugeben?

Das erste Problem ist in verteilten Systemen fundamental, aufgrund der aysnchronen Kommunikation. Daher muss festgelegt sein, welche Nachrichten zulässig sind, zu welchen Zeitpunkten eine Kommunikation erwünscht ist und wann eine Entscheidung finalisiert wird. Zu Punkt (2) muss ein Mechanismus entworfen werden, der private Informationen und Nachrichten kombiniert, um eine Vereinbarung zu treffen. Für einen solchen Mechanismus muss es ein oder mehrere Kriterien geben, anhand derer ein Schedule bewertet und verglichen werden kann. Unter diesen Annahmen lässt sich die Methodik, welche einem dezentralen Scheduling zugrunde liegt analysieren. Es ist zu bewerten in wieweit folgende Eigenschaften erfüllt sind (vgl. [41]):

- Die Föderationsteilnehmer können ohne Wissen über die Bedürfnisse und Strategien der Anderen effektive Entscheidungen treffen.
- Die Kommunikation benötigt einen nur geringen Mehraufwand.
- Eine Vereinbarung über ein Schedule ist in angemessener Zeit und vertretbarem Berechnungsaufwand möglich.
- Es werden keine Ressourcen durch eine Lösung verschwendet. Eine Lösung die für keinen Teilnehmer verbessert werden kann, ohne einen Anderen zu schädigen wird *Pareto-optimal* genannt.

Aufgrund der beschriebenen Herausforderungen des Schedulings und gewünschten Eigenschaften eines Mechanismus, werden Modelle der Wirtschaftswissenschaften genutzt um den typischen Problemen bei der Zuteilung von Ressourcen in verteilten Systemen zu begegnen (vgl. Clearwater 1996 [12] und [41]).

> May (2000) [31]: Die freie Marktwirtschaft ist eines der leistungsfähigsten und verbreitetesten Verfahren zur effizienten Allokation von knappen Ressourcen.

Davon ausgehend ergibt es Sinn, die Ergebnisse der umfangreichen Forschung in diesem Gebiet zu nutzen. Zumal die zugrunde liegenden Mechanismen passender erscheinen als traditionelle Scheduling-Methoden der Informatik, wie sie aus Ressourcen-Zuteilungsverfahren heutiger Scheduler in Arbeitsplatzrechnern bekannt sind. Diese sind meist eine zentrale Instanz in einem geschlossenen System und brauchen zur effizienten Ressourcenbelegung sämtliche, systemweite Informationen (vgl. [31]). In offenen und sich dynamisch verändernden Systemen wird eine zentrale Allokation und Koordination von Ressourcen problematisch. Freie Märkte dagegen sind von Natur aus dezentral. Alle gehandelten Güter haben ein homogenes Maß - ihre Kosten, welche sich im wesentlichen aus Angebot und Nachfrage dieses Guts bestimmen lassen. Die Akteure können effektiv anhand ihrer eigenen Bewertung eines Gutes Entscheidungen über einen Preis treffen, es ist keine zentrale Instanz nötig. Eine Vereinbarung ist einfach bei übereinstimmenden Auffassungen über den Preis zu erzielen und die Kommunikation kann im Zweifelsfall auf den Austausch von Preisvorstellungen reduziert werden. Ein Markt ist effizient, wenn die Ressourcenverteilung pareto-optimal ist, also jede Veränderung der Zuteilung zu einer Benachteiligung einer Partei führt. Dieser Begriff lässt sich gut auf die Beurteilung von Ressourcenzuordnungen durch Scheduling-Verfahren übertragen. Auf dynamische und auch unvorhersehbare Veränderungen von Angebot und Nachfrage können Marktteilnehmer durch Anpassung ihrer eigenen Einschätzung reagieren, wodurch ein Gleichgewichtszustand hergestellt werden kann, der die neue Marktsituation widerspiegelt. Auch für diese dynamische Preisfindung ist keine Steuerungsinstanz notwendig. Es werden verschiedene Preisbildungsmechanismen unterschieden, die geeignet sind einen Ausgleich von Angebot und Nachfrage und damit einen fairen Marktpreis zu bestimmen (vgl. Zahn 1999 [45]):

- Auktionen und Börsen
- bilterale Aushandlung
- Ausschreibung und Einschreibung
- Angebot- und Nachfrage-Fixierung

Für diese Arbeit wurden Scheduling-Verfahren entwickelt, die sich dem ersten oder dem letzten Mechanismus zuordnen lassen. Die Fixierung von Angebot und Nachfrage ist allgemein bekannt. Ein Anbieter legt seinen Verkaufspreis fest und ein Kunde entscheidet, ob er diesen Preis bezahlen kann und will. Auktionen werden in Abschnitt 3.4.3 eingehender vorgestellt.

3.4.1 Einordnung dieser Arbeit

Die Scheduling-Verfahren, welche im Rahmen dieser Arbeit entwickelt und bewertet werden, lassen sich sinnvoll in eine spezielle Klasse einordnen. In besonderem Fokus stehen die Möglichkeiten und Herausforderungen von Cloud-Föderationen. Die Verfahren decken die verschiedenen, hierarchischen Scheduling-Ebenen ab, wie sie in dem oben beschriebenen Modell von Kohne et al. [27] dargestellt sind. Ebenfalls dem Modell folgend, werden SLAs berücksichtigt und zwar bei der initialen Aushandlung, als auch während des gesamten Lebenszyklus einer assoziierten VM, insbesondere bei ein oder auch mehrfacher Migration. Es wird ausschließlich eine dezentral organisierte Föderation betrachtet, in der voneinander unabhängige Service-Aufträge verarbeitet werden. Im Fokus der Auswertung der Verfahren steht die System-Ebene, und dabei die Auswirkungen auf ökonomische Kennzahlen von CSPs. Es wird im weiteren Verlauf des Kapitels das für diese Ausarbeitung entworfene Finanz-Modell vorgestellt. Anhand des Modells lässt sich der Zusammenhang der Ressourcen-Nutzung und des Profits der CSPs nachvollziehen. Der Profit errechnet sich dabei aus verschiedenen Komponenten. Die heuristische Annahme der entwickelten Algorithmen ist, dass sich der Gesamtprofit durch Verbesserung seiner Teilaspekte optimieren lässt. Es werden Verfahren, die eine statische Preisbindung berücksichtigen, Auktionsverfahren mit dynamischer Preisfindung gegenübergestellt.

3.4.2 Verwandte Arbeiten

VM-Scheduling und SLAs sind unter verschiedensten Gesichtspunkten erforscht. Kalra und Singh (2015) [23] bieten hier ebenso wie Teng (2011) [40] einen Überblick über verschiedene heuristische Ansätze. Einen Fokus auf energieeffizientes Scheduling legt Sekhar et al. (2012) [37]. Patel (2009) [34] diskutiert den Einsatz von SLAs in Clouds und in dem Forschungsprojekt SLA@SOI wird eine Management-Referenzarchitektur, ein eigenes SLA-Modell und eine Monitoring-Integration vorgestellt. Cloud-Föderationen sind aktuell

Inhalt verschiedener Forschungsprojekte. *InterCloud* ist eine föderierte Cloud-Umgebung in der Applikationen ausgeführt und Veränderungen des Qualtity of Service (QoS) unter variierenden Workloads und Ressourcen ausgewertet werden können (vgl. Buyya et al. 2010 [8]). Kecskeméti et al. (2012) [24] stellen eine Architektur vor, in der verschiedene Provider über Meta-Broker und Cloud-Broker auf mehreren Ebenen und mehrere Cloud-Umgebungen verbunden sind und untersuchen den Einfluss von Policies auf den Energieverbrauch (vgl. Kecskeméti et al. 2013 [25]). Dezentrales, marktbasiertes Scheduling wird von Malone et al. 1988 [21] diskutiert. Goiri et al. (2010) [17] erörtern, basierend auf einem eigenen Finanz-Modell, in welchen Situationen Tasks lokal ausgeführt, migriert oder von Föderationspartnern angenommen werden sollten. Viele dieser Arbeiten weisen Schnittmengen zu der vorliegenden auf, umfassen aber nicht die für diese Arbeit wesentlichen Aspekte. So wird das Task-Scheduling zumeist ohne Berücksichtigung von Föderationen untersucht, in den relevanten föderierten Cloud-Umgebungen ist ein zentraler Ansatz gewählt. Die Arbeiten, welche dezentrale Organisationen berücksichtigen, beziehen keine SLAs mit ein. Die behandelten Finanz-Modelle dienen zur Bewertung einer Situation, werden aber nicht zur Entscheidungsfindung des Schedulings herangezogen.

3.4.3 Auktionsbasiertes Scheduling

Auktionen sind ein Mechanismus, bei dem konkurrierende Gebote verglichen und daraus resultierend Preise und die Zuteilungen von Gütern bestimmt werden (vgl. Wolfstetter 1998 [44]). Entscheidend sind explizit festgesetzte Regeln, die sowohl die Kommunikation der Teilnehmer, als auch im Besonderen die Festlegung von Preis und Zuschlag bestimmen. Nur durch diese fixen Regeln unterscheiden sie sich von informellen Verhandlungen. Es werden Auktionsformen unterschieden nach offenen und verdeckten Geboten. Bei offenen Geboten können Bieter auf die bekanntgegebenen Preise eines Konkurrenten reagieren. Bei verdeckten Geboten bleiben die Preise der Teilnehmer unzugänglich. Für jede dieser zwei Klassen gibt es zwei wichtige Repräsentanten, die sich in Ablauf und Preisbestimmung unterscheiden. Nahezu alle weiteren Auktionsformen lassen sich als eine Mischform aus diesen vier wichtigsten Formen darstellen.

Englische Auktionen

Die Gebote der Teilnehmer sind öffentlich einzusehen. Es kann sukzessive ein Preis erhöht werden, bis ein Zuschlag erteilt wird.

Holländische Auktionen

Diese Auktionsform gleicht der englischen Variante, unterscheidet sich allerdings darin, dass die Preisforderung des Auktionators mit der Zeit abnimmt. Der erste Teilnehmer, welcher bereit ist den geforderten Betrag zu zahlen, erhält den Zuschlag.

Höchstpreis-Auktionen

Durch verdeckte Gebote können Teilnehmer ihre Wertschätzung abgeben. Es werden unter Umständen mehrere Runden von Geboten akzeptiert. Nach Abschluss der Bietrunden erhält das Höchstgebot den Zuschlag.

Vickrey-Auktionen

Diese spezielle Form verläuft analog zu einer Höchstpreis-Auktion. Allerdings zahlt der Gewinner das zweit-höchste Gebot statt seines Eigenen.

Für Scheduling-Verfahren bieten Auktionen die Möglichkeit in einem dezentralen Markt eine ökonomisch effiziente Zuteilung durchzuführen. Effizient bedeutet in diesem Zusammenhang, dass ähnlich einer zentralen Scheduling-Instanz eine Zuteilung die globale Nutzenfunktion maximiert. Dazu muss in einem markt-basierten Scheduling die Belegung gewählt werden, mit welcher der höchste Gewinn zu erzielen ist. Dazu müssen die Teilnehmer einer Auktion allerdings entsprechend ihrer wahren, privaten Wertschätzung bieten. In dezentralen Scheduling-Verfahren werden die Entscheidungen auf den wenigen vorhandenen Informationen getroffen. Auktionen ermöglichen den Teilnehmern Informationen offenzulegen, die das Scheduling positiv beeinflussen. Um strategisches Bieten in Relation zu Konkurrenzgeboten zu verhindern, sind verdeckte Auktionsformen besser geeignet, während bei offenen Geboten die Wahrscheinlichkeit höher ist den Käufer mit dem höchsten Nutzen zu identifizieren (vgl. [31]).

3.4.4 Finanzmodell

Ein Ziel dieser Arbeit ist es, Auswirkungen von Scheduling-Verfahren auf ökonomische Kennzahlen von CSPs zu untersuchen. Um diese Kennzahlen zu erhalten ist ein Finanz-Modell nötig. Die Zusammenhänge von messbaren Vorgängen in Cloud-Infrastrukturen und daraus resultierenden monetären Größen sollen dargestellt werden. Für diese Arbeit wurde ein Modell entworfen, dass als Entwurfsgrundlage für die Erweiterung des FederatedCloudSim-Frameworks dient. Diese Erweiterung ermöglicht es Cloud-Infrastrukturen und Scheduling-Strategien zu simulieren und die Einnahmen und den Profit zu berechnen und dabei SLAs zu berücksichtigen. Eine zweite Anwendung des Finanz-Modells besteht darin, Zusammenhänge zu identifizieren und Möglichkeiten zu offenbaren, die zur Entscheidungsfindung der Scheduling-Verfahren beitragen können.
Um zusammen mit dem Modell für SLA-basiertes VM-Scheduling eingesetzt werden zu können, ist das Modell sehr variabel gestaltet. Es können vielfältige Cloud-Infrastrukturen abgebildet werden, von einem Provider mit einem Rechenzentrum, bis zu einer Föderation von Providern mit beliebig vielen Rechenzentren. Es erlaubt zudem eindeutige Kostenstellen zu benennen, die für Einnahmen bzw. Ausgaben verantwortlich sind. Abbildung 3.2 zeigt die gewählte hierarchische Struktur. Von den verursachenden Kostenstellen aus, welche auf der untersten Ebene dargestellt sind, hin zu den finanziellen Kennzahlen lässt

sich der Zusammenhang erkennen, wie die Komponenten aggregiert werden. Im Zentrum stehen die drei abstrakten Metriken für Einnahmen, Kosten und Strafen, die für jede Cloud-Infrastruktur mit Berücksichtigung von SLAs relevant sind. Nach unten hin ist ersichtlich aus welchen Quellen heraus sie sich zusammensetzen, nämlich die Einnahmen durch Ausführung von VMs im Rahmen eines Service, der für einen Kunden erbracht wird und durch Ausführung von VMs, die in einer Föderation von einem Partner übernommen wurden. Dieser zahlt hierfür eine Entschädigung, die der ausführende Provider als Eingang verbuchen kann. Für die Berechnung der Kosten werden die Gebühren berechnet, die für die Auslagerung von VMs in eine Föderation anfallen, die operationalen Ausgaben (englisch: Operational Expenditure (Opex)), die für den Betrieb der Infrastruktur anfallen und den Investitionskosten (englisch: Capital Expenditure (Capex)), die für Anschaffung, Erneuerung oder Verbesserung der Infrastruktur bezahlt werden. Obwohl Investitionskosten nicht in festen Zyklen anfallen, werden sie zur internen Abrechnung als monatliche Kosten für jede Servereinheit über ihre erwartete Lebensdauer verbucht (vgl. Cisco 2010 [1]). Die Strafen wiederum ergeben sich aus Verletzungen der in SLAs festgelegten SLOs. Dabei wird zwischen Strafen unterschieden, die gegenüber einem Kunden anfallen und solchen die an Föderationspartner zu zahlen sind. Von Einnahmen, Kosten und Strafen aus nach oben werden diese Kennzahlen zusammengezogen, um den Profit zu berechnen. Sei n die Anzahl der modellierten Rechenzentren und m die Anzahl der CSPs. Je nach Komplexität der Struktur kann aus den Profiten der Rechenzentren mit

$$DataCenterProfit = DataCenterEinnahmen - DataCenterKosten - DataCenterStrafen$$

im Weiteren auch der Profit der CSPs

$$CSPProfit = \sum_{i=1}^{n} (DataCenterProfit)_i$$

und schließlich der Föderation ermittelt werden

$$FederationProfit = \sum_{i=1}^{m} (CSPProfit)_i$$

Mit der zusätzlichen Darstellung von Sender und Empfänger sind die beschriebenen Geldflüsse auch in den Abbildungen 3.3 und 3.4 gezeigt. Im der ersten ist der allgemeine Fall gezeigt, dass ein Provider einen Service für einen Kunden ausführt und dazu die Infrastruktur bereitstellt. Die zweite Abbildung zeigt den Fall einer migrierten VM, welche möglicherweise innerhalb ihres Lebenszyklus bei verschiedenen Föderationspartnern ausgeführt wird. Hier wird auch die Situation verdeutlicht, dass die SLAs zwischen CSP und Kunde andere Kennzahlen aufweist als eine SLA zwischen CSPs. Dann kann es vorkommen, dass ein Provider effektiv mehr oder weniger Einnahmen von einem Kunden generiert, als er für die Ausführung in der Föderation zahlen muss. Analog können auch effektiv höhere oder niedrigere Strafzahlungen an einen Kunden entstehen als durch Zahlungen aus der

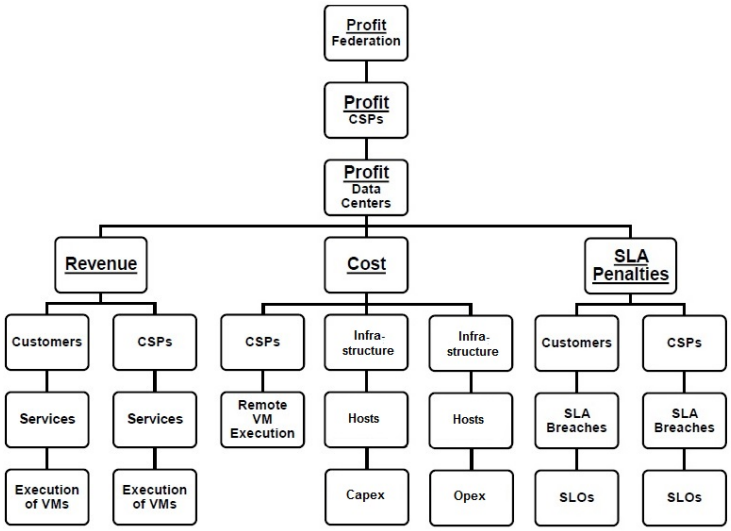

Abbildung 3.2: Finanz Modell

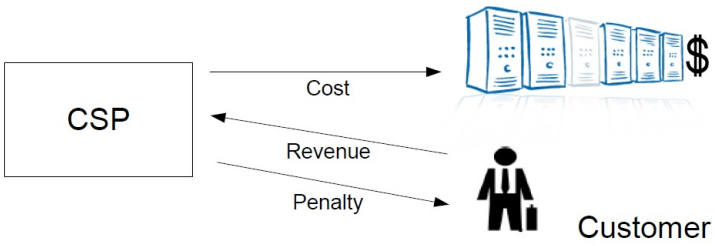

Abbildung 3.3: Modell des Geldflusses im allgemeinen Fall

Föderation aggregiert werden. Zusammenfassend lässt sich festhalten, dass alle für diese Arbeit relevanten, finanziellen Metriken durch dieses Modell abgedeckt sind und es aber gleichzeitig so konzipiert ist, dass zusätzliche Quellen und auch Geldströme eingehangen werden können. Zudem ist klar zu erkennen, dass zur Optimierung des Profits der Föderation der Profit der CSPs bzw. Rechenzentren verbessert werden sollte. Das sich der Profit aus Einnahmen, Kosten und Strafen zusammensetzt und damit drei Ansatzpunkte bietet um positiven Einfluss zu nehmen. In Abschnitt 4.4.5 werden daher die Scheduling-Strategien besprochen, welche explizit versuchen an einem oder mehreren Punkten Einfluss zu nehmen. Die Basis dazu bieten die während der Simulation berechenbaren Kennzahlen, die anhand dieses Modells ermittelt werden.

3.4.5 Vertrauenswert

Anhand der Metriken, die auf Basis des Finanz-Modells zur Verfügung stehen können Scheduling-Entscheidungen getroffen werden. In dem Diagramm zum Finanz-Modell lässt sich gut erkennen welche Kostenstellen zur Verbesserung der Einnahmen, Kosten und Stra-

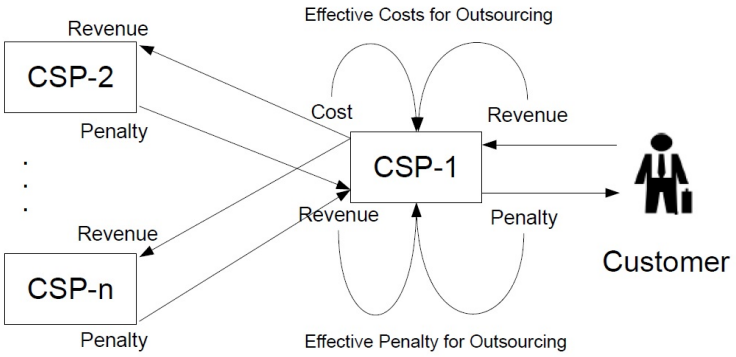

Abbildung 3.4: Modell des Geldflusses im Fall von föderierten VMs

fen zu beeinflusst sind. Um insbesondere die Wirkung der Strafen steuern zu können, wird im Folgenden eine Metrik entworfen. Es wird ein Vertrauenswert berechnet, der die Zuverlässigkeit von Föderationspartnern in Bezug auf ihr Verhalten hinsichtlich der SLAs bewertet. Um zu beurteilen, ob Föderationspartner in der Lage sind die Vereinbarungen der SLAs einzuhalten, werden als Eingabedaten die bisherigen SLA-Brüche und die Ausfallzeiten der zu einem Provider migrierten VMs verarbeitet. Der Vertrauenswert soll prozentual zum Ausdruck bringen, wie zuverlässig ein Föderationsteilnehmer entgegengenommene Aufträge handhabt und liegt damit zwischen 0% und 100%. Ein CSP kann diese Metrik für jeden seiner Partner berechnen und sie damit untereinander bezüglich ihrer Zuverlässigkeit vergleichen.
Das Ergebnis wird aus zwei Teilberechnungen zusammengezogen. Im ersten Teil werden die verursachten SLA-Brüche, im zweiten Teil die verschuldeten Ausfallzeiten beurteilt. Für die Beurteilung der SLA-Brüche wird für jede migrierte VM geprüft, ob eine Verstoß vorgefallen ist. Kontrolliert werden Verletzungen der Grenzen der:

- Prozessorauslastung
- Arbeitsspeicherversorgung
- Erreichbarkeit der VM

Ist eine der drei Unregelmäßigkeiten aufgetreten, so wird diese VM mit einem Bruch als nicht erfolgreich behandelt markiert. Sei n die Anzahl der migrierten VMs. Weiterhin gebe $p_{i,j} \in \{0, 1, \ldots, k\}$ an, welchem der k Provider die i-te VM im j-ten Zeitraum mit $j \geq 1$ zugeordnet ist. Und es sei $b_{i,j} = 1$, falls es zu einem SLA-Bruch kam und ansonsten wurde VM_i zum Zeitpunkt j erfolgreich verarbeitet und der Wert sei 0. Falls die i-te VM zum Zeitpunkt j keinem Provider zugeordnet ist, sei $b_{i,j} = 0$.

Damit ergibt sich der prozentuale Wert für SLA-Brüche für Provider ℓ zum Zeitpunkt t durch:

$$b_{\ell,t} = \frac{\sum_{i=1}^{n} \begin{cases} 1, \text{falls } p_{i,t} = \ell \text{ und } b_{i,t} = 1 \\ 0, \text{sonst} \end{cases}}{\sum_{i=1}^{n} \begin{cases} 1, \text{falls } p_{i,t} = \ell \\ 0, \text{sonst} \end{cases}}$$

Sollte dieser Wert nicht definiert sein, weil der Nenner Null ist, sei sinnvollerweise $b_{\ell,t} = 0$ definiert, gleiches gilt für die im Folgenden definierten Quotienten. Neben dieser ersten Berechnung auf Grundlage der SLA-Brüche werden im zweiten Teil die Ausfallzeiten, die ein Provider verschuldet hat, bewertet. Sei $a_{i,j} = 1$, falls es zu einer Ausfallzeit kam und ansonsten sei der Wert 0. Durch die Dienstgüte eines SLA ist eine Grenze z (mit $0 < z \leq 1$ und etwa $z = 0.99$) festgelegt, welche die minimal zu tolerierende Erreichbarkeit beschreibt. Zur Beurteilung der Ausfallzeiten wird die SLA-Gefährdung als prozentualer Anteil der angefallenen Ausfallzeiten einer VM an der laut Dienstgüte maximal erlaubten Zeit definiert durch

$$sg_{i,\ell,t} = \frac{\sum_{j=1}^{t} \begin{cases} 1, \text{falls } p_{i,j} = \ell \text{ und } a_{i,j} = 1 \\ 0, \text{sonst} \end{cases}}{z \cdot \sum_{j=1}^{t} \begin{cases} 1, \text{falls } p_{i,j} = \ell \\ 0, \text{sonst} \end{cases}} \quad .$$

Außerdem wird der Anteil, der bei einem Provider angefallenen Ausfallzeit an der gesamten Ausfallzeit einer VM definiert durch

$$aa_{i,\ell,t} = \frac{\sum_{j=1}^{t} \begin{cases} 1, \text{falls } p_{i,j} = \ell \text{ und } a_{i,j} = 1 \\ 0, \text{sonst} \end{cases}}{\sum_{l=1}^{k} \sum_{j=1}^{t} \begin{cases} 1, \text{falls } p_{i,j} = l \text{ und } a_{i,j} = 1 \\ 0, \text{sonst} \end{cases}}$$

Eine Ausfallzeit-Verletzung $av_{i,\ell,t} = 1$ liegt vor, wenn sowohl $sg_{i,\ell,t}$ als auch $aa_{i,\ell,t}$ größer als vordefinierte Grenzen sind. Föderationspartnern wird somit ein Spielraum gewährt, da nicht jede Ausfallzeit negativ zu werten ist. Beispielsweise kann eine Fehlzeit aufgrund einer sinnvollen Migration zwischen Hosts durchaus erwünscht sein. Ebenso soll ein Partner nicht schlecht bewertet werden, falls sein Beitrag zu der Gefährdung einer VM zu klein ist. Sollte eine VM auf einem akzeptablen Gefährdungsniveau liegen oder der Anteil des Provider an den bisherigen Fehlzeiten gering genug sein, so gilt eine VM als erfolgreich behandelt. Die prozentuale Ausfallzeit-Verletzung für Provider ℓ zum Zeitpunkt t ergibt sich durch:

$$v_{\ell,t} = \frac{\sum_{i=1}^{n} \begin{cases} ab_{i,\ell,t}, \text{falls } p_{i,t} = \ell \text{ und } a_{i,t} = 1 \\ 0, \text{sonst} \end{cases}}{\sum_{i=1}^{n} \begin{cases} 1, \text{falls } p_{i,t} = \ell \\ 0, \text{sonst} \end{cases}}$$

Die Zuverlässigkeit eines Provider ℓ zum Zeitpunkt t in Bezug auf SLA-Brüche ergibt sich durch

$$zb_{\ell,t} = 1 - b_{\ell,t}$$

und analog die Zuverlässigkeit bezüglich der Ausfallzeit-Verletzungen durch

$$zv_{\ell,t} = 1 - v_{\ell,t} \quad .$$

Die beiden prozentualen Teilergebnisse werden durch zwei Parameter gewichtet, sodass entweder den SLA-Brüchen oder den Ausfallzeiten mehr, weniger oder gleich viel Bedeutung zugesprochen wird und es ergibt sich der Vertrauenswert eines Provider ℓ zum Zeitpunkt t durch

$$vw_{\ell,t} = q_v \cdot zb_{\ell,t} + q_w \cdot zv_{\ell,t}$$

mit $q_v, q_w \geq 0$ und $q_v + q_w = 1$. Um die Zuverlässigkeit eines Partners nicht nur im letzten betrachteten Zeitraum, sondern über die gesamte Partnerschaft zu betrachten, wird eine exponentielle Glättung ausgeführt durch

$$vw^{\text{final}}_{\ell,t} = \alpha \cdot vw_{\ell,t} + (1 - \alpha) \cdot vw^{\text{final}}_{\ell,t-1}$$

mit $vw^{\text{final}}_{\ell,0} = vw_{\ell,0}$ und einem Glättungsfaktor $\alpha \in (0, 1)$. Als explizite Formel erhalten wir somit

$$vw^{\text{final}}_{\ell,t} = \sum_{j=0}^{t-1} (1 - \alpha)^j \alpha \cdot vw_{\ell,t-j} + (1 - \alpha)^t vw^{\text{final}}_{\ell,0} \quad .$$

Dadurch können Werte der jüngeren Vergangenheit stärker oder weniger stark gegenüber weit zurückliegenden Werten gewichtet werden. Die Gewichtung erfolgt über den Glättungsparameter α, wobei ein großes α eine starke Gewichtung der Gegenwart, ein kleines α eine höhere Gewichtung vergangener Werte bedeutet (vgl. Schneider und Mentemeier 2016 [36]).
Da diese Metrik in dezentralen Verfahren verwendet wird, kann ein Provider sie jeweils für jeden CSP seiner Föderation basierend auf Information berechnen, die ihm zur Verfügung stehen. Das sind in diesem Fall, die SLA-Brüche und Ausfallzeiten seiner VMs,

welche zu einem Partner migriert wurden. Unregelmäßigkeiten auf fremden VMs bleiben unberücksichtigt.

Kapitel 4

FederatedCloudSim

In diesem Kapitel wird zuerst in Abschnitt 4.1 das Framework *CloudSim* vorgestellt. Es dient als Basis für die Erweiterungen, die der Cloud-Simulator *FederatedCloudSim (FCS)* bietet, welche in Abschnitt 4.2 beschrieben werden. Im sich anschließenden Abschnitt 4.3 werden die Anforderungen analysiert, die sich für die Umsetzung der Scheduling-Verfahren und nötigen Framework-Erweiterungen im Rahmen dieser Arbeit ergeben, bevor in Abschnitt 4.4 die erforderlichen Verfahren und Algorithmen entwickelt werden und ihre Umsetzung in Abschnitt 4.5 gezeigt wird.

4.1 CloudSim

CloudSim wurde von Calheiros et al. (2011) [11] entwickelt und ist ein Programm zur Modellierung und Simulation von Cloud-Infrastrukturen und Cloud-Services. Das in Java programmierte Tool stellt Klassen für alle relevanten Komponenten eines CSP zur Verfügung. Berücksichtigt sind die physikalischen Komponenten, wie Datenzentren, Hosts und Netzwerkelemente, ebenso wie Cloudlets, womit die einzelnen Jobs eines Service bezeichnet werden, sowie VMs, welche diese verarbeiten. Es gibt eine standardisierte Möglichkeit zur Entgegennahme von Workloads und eine Auswahl an Verfahren zum dynamischen Management der Ressourcennutzung, des Schedulings auf Hostebene, sowie zur energieeffizienten Auswahl und Distribution von VMs. Die CloudSim-Architektur ist unterteilt in drei Ebenen (siehe Abbildung 4.1). Die unterste Ebene ist für den Simulationsablauf verantwortlich, sie enthält die Simulationskomponenten und unterstützt ihre Interaktion. In der CloudSim-Ebene sind die Komponenten der Cloud-Infrastruktur angesiedelt und es sind die Funktionen spezifiziert, welche die Ausführung der Cloud-Services ermöglichen. Die oberste Ebene ist zur Unterstützung des Anwenders gedacht. Hier können der anwenderspezifische Infrastrukturaufbau, Algorithmen und Konfigurationen eingebunden werden.

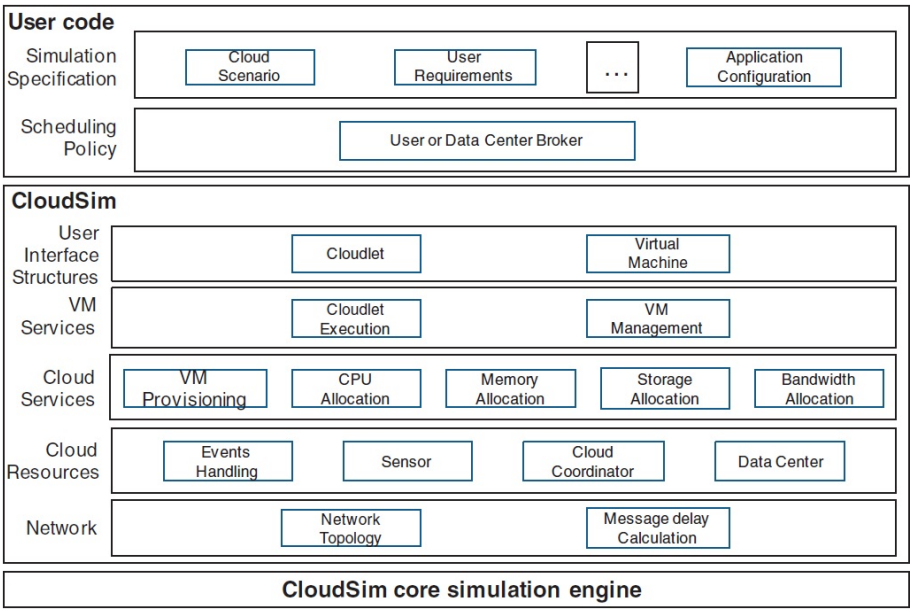

Abbildung 4.1: CloudSim-Architektur (vgl. Calheiros 2011 [11], Figure 3)

Das Framework setzt eine ereignisbasierte Simulation um. Dies bedeutet, dass zu festgelegten Zeitpunkten alle Ereignisse, welche durch *SimEvent-Objekte* repräsentiert werden und in einer entsprechenden Warteschlange gelistet sind, abgearbeitet werden und die Statusübergänge auslösen, die im Modell im zurückliegenden Simulationsschritt aufgetreten sind. Einzelne simulierte Komponenten erben von der abstrakten Klasse `SimEntity` und werden so zu Simulationseinheiten, die in der Lage sind Nachrichten zu versenden und zu empfangen, an denen Simulationsereignisse angehängt sind (siehe Abbildung 4.2). Jede Nachricht kann mit einer Verzögerung versehen werden, wodurch es möglich ist, die zeitlichen Auswirkungen eines Ereignisses zu bestimmen. Alle Nachrichten werden in eine Warteschlange, die `FutureQueue`, sortiert nach dem Zeitpunkt ihres Eintretens eingereiht. Wird ein Simulationsschritt ausgeführt, werden alle Nachrichten, die in dem damit übersprungenen Zeitraum aufgetreten sind in die `DeferredQueue` verschoben und bearbeitet. Dadurch ist es möglich, indem Nachrichten ohne Verzögerung verschickt werden, Ereignisse direkt in die aktuelle Verarbeitungsliste einzuhängen und damit Statusveränderungen zu bewirken.

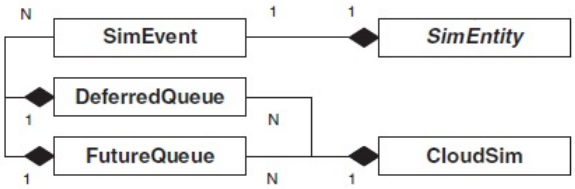

Abbildung 4.2: Klassendiagramm des CloudSim-Simulationskern (vgl. Calheiros 2011 [11], Figure 7a)

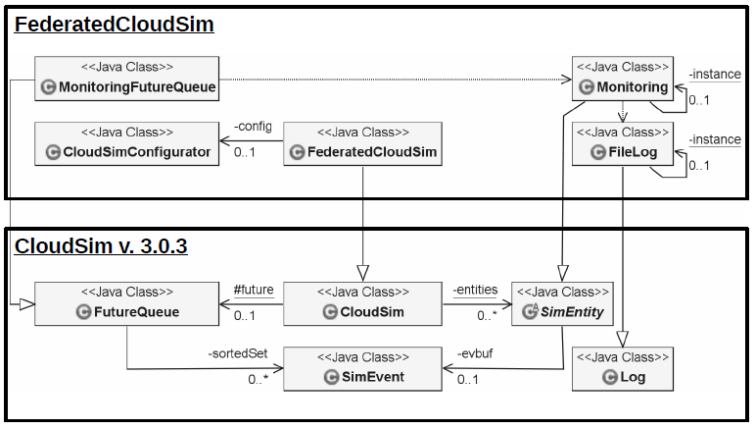

Abbildung 4.3: Erweiterung des Simulationskerns in FederatedCloudSim (vgl. KOHNE 2014 [28], Figure 4)

4.2 Erweiterungen durch FederatedCloudSim

CloudSim wird durch das von KOHNE et al. (2014) [28] vorgestellte Simulationsframework *FederatedCloudSim* um zahlreiche Funktionen erweitert. Im Wesentlichen ist dies die Modellierung von Cloud-Föderationen. Als Grundlage der Migration in der Föderation werden SLAs berücksichtigt. Sie könne als Verträge sowohl zwischen einem Provider und seinem Kunden, als auch zwischen Providern modelliert werden. Zudem wird eine komfortable Konfiguration des Modells über eine XML-Schnittstelle und ein effektives Monitoring der Ressourcen mit umfangreichem Loggen der Simulationsergebnisse ermöglicht. Abbildung 4.3 zeigt, dass die Erweiterung des Simulationskerns durch geschickte Vererbungsbeziehung den CloudSim-Code nicht verändert. Die Klasse `CloudSimConfigurator` ist für das Auslesen der Konfiguration aus einer XML-Datei zuständig. In dieser kann die gewünschte Cloud-Infrastruktur mit Anzahl und Eigenschaften der Host, Rechenzentren und CSPs gestaltet werden, dazu gehören sowohl die Ausgestaltung der modellierten Hardware, als auch die von den Providern genutzten Strategien, ihre Services und Service-Level-Agreements. In einer adäquaten XML Schema Definition (XSD)-Datei sind diese XML-Strukturen mit ihren notwendigen und optionalen Elementen und Attributen definiert. Neben den wiederverwendbaren Vorlagen der XML-Datei verwendet der `CloudSimConfigurator` zusätzlich eine Properties-Datei, in welcher die für die Simulation notwendigen Grundkonfigurationen, wie die Pfade zum Workload und zur Konfigurationsdatei, ebenso wie Logging-Optionen und Konstanten von Simulations-Parameter festgesetzt werden. Anhand der `MonitoringFutureQueue` werden die Ereignisse, welche für das Monitoring relevant sind erfasst und schon zur Simulationszeit geloggt.

Die ausgesprochen variablen Gestaltungsmöglichkeiten von Cloud-Föderationen und den darin entstehenden strategischen Entscheidungspunkten stellen die relevanteste Funktionsbereicherung des Frameworks dar. Abbildung 4.4 dokumentiert wie die neue Klas-

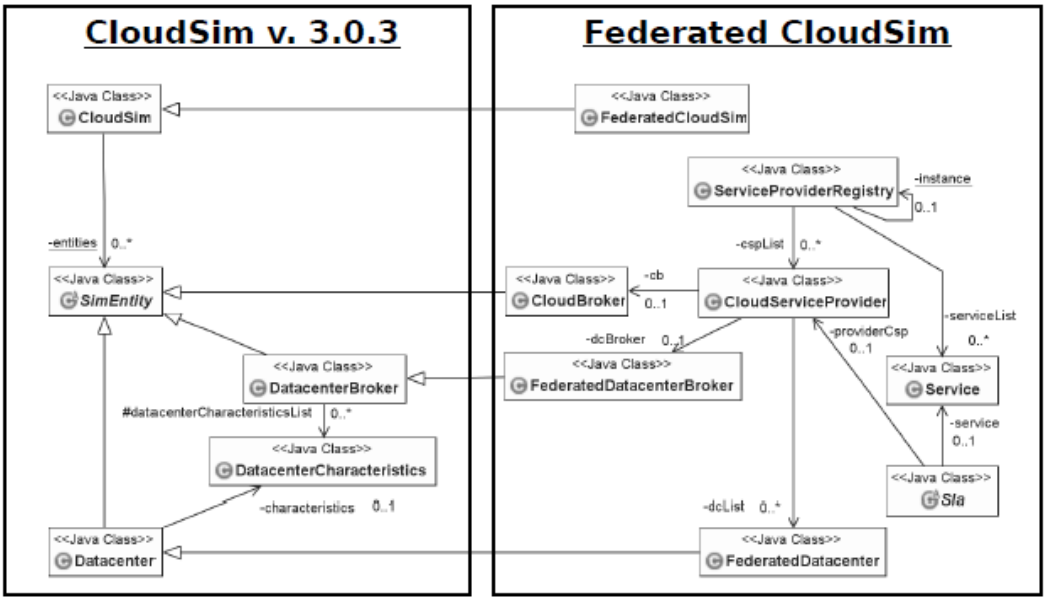

Abbildung 4.4: Struktur des FederatedCloudSim-Frameworks (vgl. KOHNE 2014 [28], Figure 3)

senstruktur durch Vererbungshierarchie auf den CloudSim-Klassen aufbaut. Es können nun CSPs simuliert werden, die verschieden ausgestaltete Rechenzentren kontrollieren. Das in Abschnitt 2.3 vorgestellte Multi-Level-Scheduling wird von FCS unterstützt. Im Klassendiagram (siehe Abbildung 4.5) sind die drei verschiedenen Ebenen (Datacenter, CSP, Föderation) dargestellt. Auf jeder Ebene wird mit einer eigenen, unabhängigen Strategie Einfluss auf die Scheduling-Verfahren genommen. Durch die Nutzung des Strategie-Entwurfs-Muster können Nutzer des Simulations-Frameworks auf jeder Ebene eigene Strategien entwerfen und testen oder sogar zur Laufzeit die Strategie wechseln (vgl. GOLL 2014 [18], S.179). Aus dem CloudSim-Framework verfügt jedes `FederatedDatacenter` über eine `VmAllocationPolicy`, welche die Zuteilung von VMs zu physikalischen Hosts koordiniert. Jeder CSP besitzt einen `FederatedDatacenterBroker`, der die zentrale Instanz darstellt, in der alle Information eines Providers zusammengeführt werden, die für das Scheduling relevant sind. Basierend auf den Kenntnissen zu angebotenen Diensten und Ressourcenauslastung aller angeschlossenen Rechenzentren wird eine `FederatedDatacenterBrokerStrategy` zur Auswahl und Platzierung von Virtuellen Maschinen umgesetzt. Der `CloudBroker` stellt die Kommunikationsschnittstelle zu föderierten CSPs dar und dies beinhaltet ebenfalls die Umsetzung der strategischen Entscheidungen auf dieser Ebene. Dazu besitzt er eine `CloudBrokerStrategy`. FederatedCloudSim weist zudem ein Grundgerüst zur Erstellung und Verwaltung von Services und SLAs auf. Durch den `CloudSimConfigurator` werden aus den eingelesenen Workloads und der XML-Konfigurationsdatei SLAs zu den Service-Requests und SLAs zwischen Providern erstellt. Der `SlaManager` eines CSP ist im Simulationsverlauf für die Verwaltung der SLAs zuständig, dazu kontrolliert er in jedem

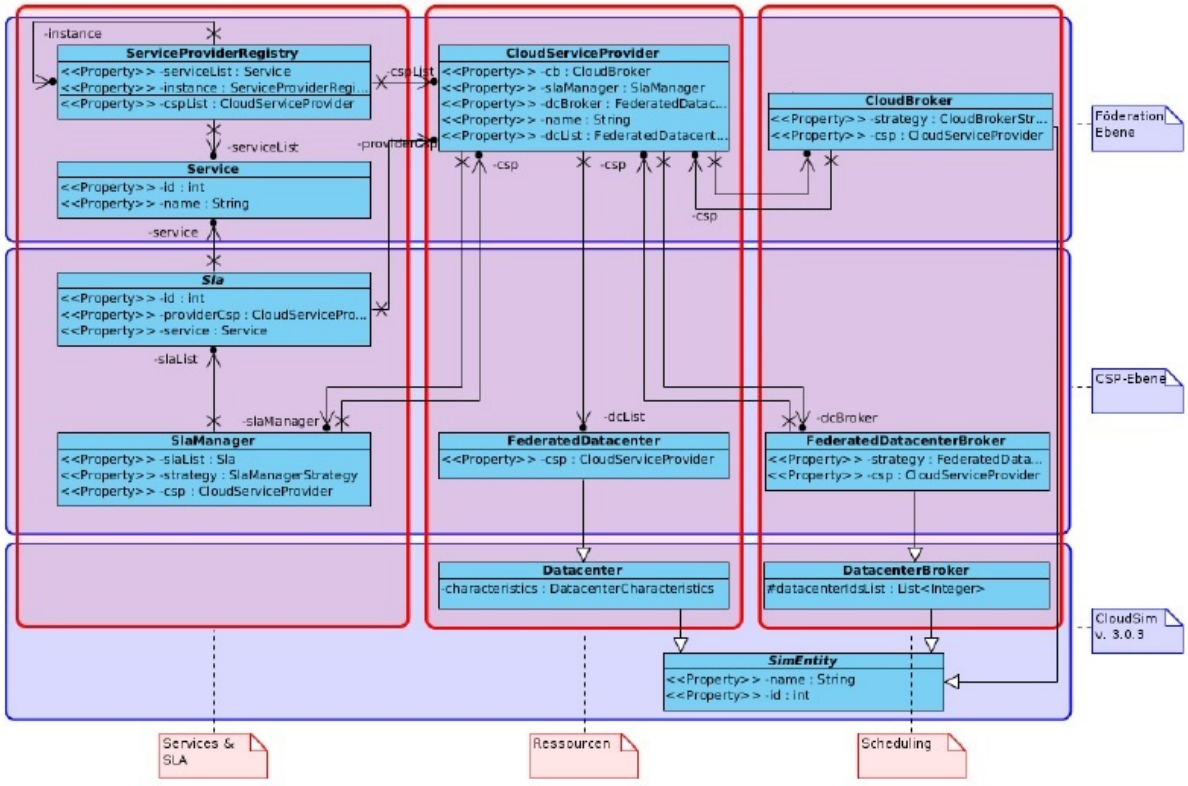

Abbildung 4.5: Klassendiagram zu den wesentlichen Erweiterungen durch FederatedCloudSim (vgl. SPOHR 2015 [38], Abbildung 4.4)

Simulationsschritt, ob in einem Rechenzentrum eine SLA-Gefährdung aufgetreten ist und initialisiert im positiven Fall die Migration von VMs, um diese Gefährdung aufzulösen.

4.3 Analyse der notwendigen Erweiterungen

Im Abschnitt 3.4.4 wurde beschrieben, wie ein Finanzmodell als Grundlage für Scheduling-Verfahren dienen kann und in Abschnitt 3.4.3 wurde die Möglichkeit erörtert, Auktionen in Scheduling-Entscheidungen mit einzubeziehen. Um die Einflüsse und Auswirkungen dieser Verfahrensvarianten untersuchen zu können, wird der Simulator FederatedCloudSim um diverse Funktionen ergänzt. Diese Erweiterungen und die Anforderungen, welche dabei erfüllt werden sollen, werden nachfolgend beschrieben:

- *Finanzmodell*: Der Anwender erhält die Möglichkeit, die Auswirkungen unterschiedlicher Cloud-Infrastrukturen und verschiedener Scheduling-Verfahren auf den wirtschaftlichen Ertrag simulieren zu können. Dazu muss das Framework die Möglichkeit bieten, vordefinierte oder anwenderspezifische Abrechnungsmodelle einzubinden und vom Anwender angegebene Parameter zur Berechnung einzulesen.

- *Service Level Agreements*: Eine wesentliche Grundlage der Berechnung von wirtschaftlichen Erträgen bilden, wie in Abschnitt 3.4.4 beschrieben, die zwischen Kunde und Anbieter, sowie zwischen Anbieter und professionellem Nutzer ausgehandelten Entgelte für die Nutzung der Services und Strafzahlungen für die Nichteinhaltung der Servicegarantien. Da das Framework, insbesondere auf Föderationsebene, für die Gestaltung der SLAs nur technische Vereinbarungen berücksichtigt, ist dieses Konzept um finanzielle Vereinbarungen zu erweitern, die folglich zur Simulationszeit berücksichtigt werden können.

- *Scheduling-Eventfluss*: Ereignisse, die einen Einfluss auf die simulierte Zeit haben können, werden im FederatedCloudSim durch Nachrichten zwischen Simulationseinheiten realisiert (vgl. Abschnitt 4.1). Die Migration von VMs erfolgt infolgedessen durch Nachrichten an die Schedulinginstanz der gewünschten Ebene. Das ist entsprechend der Scheduler eines Rechenzentrums, der `FederatedDatacenterBroker` oder der `CloudBroker`. Es ist möglich, den Nachrichten in einem Datenfeld eine Datenstruktur mit Referenzen auf die relevanten VMs anzuhängen. FederatedCloudSim unterstützt den Nachrichtenfluss zur initialen Verteilung von Service-Requests, zur Migration Virtueller Maschinen auf Host-Ebene innerhalb eines Rechenzentrums und zur Verschiebung von Virtuellen Maschinen zwischen Rechenzentren eines Service-Providers. Für die Migration einer VM zu einem föderierten Provider muss der `CloudBroker` um eine umfassende Ereignisbehandlung erweitert werden, die alle Migrationsentscheidungsvarianten erfasst. Dabei müssen Schnittstellen für die Scheduling-Strategie angeboten werden, an denen der Provider strategische Entscheidungen über Annahme und resultierendem Scheduling oder Ablehnung von Anfragen bestimmen kann. Um die Szenarien vollständig behandeln zu können, die aus Ablehnungen von Migrationsanfragen und Rückmigrationen entstehen können, muss ebenso der `FederatedDatacenterBroker` um sinnvolle Routinen ergänzt werden.

- *Auktionsverfahren*: Scheduling-Verfahren, welche die Ergebnissen einer Auktion mit einbeziehen sind eine neue Funktion des Simulations-Framework. Es ist eine Klassenstruktur zu entwerfen, mit der verschiedene Auktionsformen realisierbar sind. Die Durchführung einer Auktion muss in den Nachrichtenfluss zwischen den Kommunikationseinheiten der Provider eingebunden werden. Ebenso sind die Ergebnisse der Auktionen in ihren Auswirkungen auf die SLAs zu berücksichtigen.

- *Scheduling-Strategien*: Basierend auf dem Multi-Level-Scheduling-Modell werden Ressourcen-Management und Distributions-Entscheidungen durch Strategien auf drei verschiedenen Ebenen realisiert. Zur Umsetzung des Ziel finanzielle Metriken durch Scheduling zu verbessern werden auf allen Ebenen neue Strategien implementiert.

Um den Einfluss von SLAs auf die Kapitalströme bewerten zu können, wird das Framework um eine Komponente zur Berechnung einer Vertrauens-Metrik erweitert.

4.4 Entwurf

Zu den in der Analyse beschriebenen Anforderungen werden in diesem Abschnitt die Konstruktion der erweiterten Systemarchitektur, der Subsysteme, die Komponenten und Algorithmen erläutert.

4.4.1 Finanzmodell

Jeder CSP soll aktuelle, detailierte Informationen über die finanziellen Metriken seines Systems abrufen können. Der Nutzer des Simulators braucht die Möglichkeit, sein Modell zur Berechnung dieser Metriken einbinden und mit Parametern konfigurieren zu können. In Abbildung 4.6 ist im linken Block die hierarchische Struktur der Cloud-Komponenten, wie sie in FCS implementiert sind, wiedergegeben. Zur Initialisierung der Simulation werden die CSPs und ihre Struktur an Rechenzentren und enthaltenen Hosts durch den `CloudSimConfigurator` erzeugt. Ebenso werden die workload-spezifischen VMs generiert und allokiert. Der rechte Block umfasst die Strukturen zur Berechnung der Wirtschafts-Metriken. Jeder CSP erhält dabei eine `Accounting`-Komponente. Im Laufe der Simulation wird in jedem Schritt die zentrale Methode `update()` dieses Objekts aufgerufen, wodurch die Kosten und Einnahmen des Providers aktualisiert und über passende Schnittstellen abgefragt werden können. In welcher Weise die Berechnung zu erfolgen hat, wird durch eine vom Nutzer zu implementierende Strategie vorgegeben. Es muss dabei die abstrakte Klasse `AccountingStrategy` erweitert werden, durch welche die grundlegenden Funktionen zur Berechnung von Kosten und Einnahmen fixiert sind. In dem dargestellten Klassendiagramm ist beispielhaft die `AccountingStrategyBasic` eingebunden. Sie implementiert das in 3.4.4 beschriebene Finanzmodell, das als Objekt zur Kalkulation die VMs und Hosts eines CSPs nutzt. Aufgrund des Strategie-Entwurfsmusters können verschiedenste Berechnungsmodelle eingesetzt werden.

4.4.2 SLA und Finanz-Parameter

Um neben dem Abrechnungsmodell auch über die XML-Konfigurationsdatei die Parameter, beispielsweise die Kosten zur Nutzung einer Ressource, konfigurieren zu können, wird die XSD-Datei erweitert. Jeder CSP erhält ein zusätzliches Element, in dem alle für ein Modell nötigen Parameter gesetzt werden können. Diese Werte werden bei der Initialisierung der Cloud-Infrastruktur ausgelesen und gespeichert. Sie bilden die Grundlage zur Berechnung des Preises einer Service-Leistung im Falle eines Service-Requests. Ebenso werden die gleichen Parameter, mit gegebenenfalls anderer Wertbelegung, in der Schema-Definition

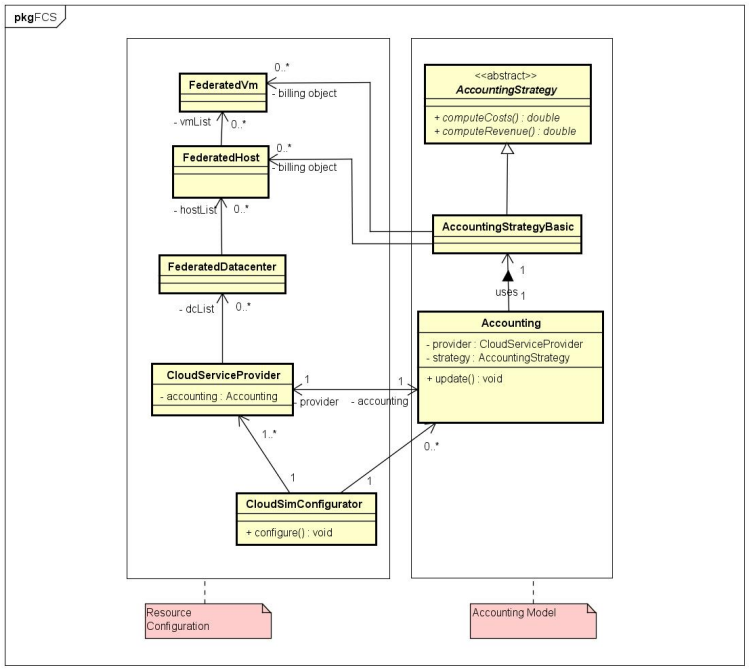

Abbildung 4.6: Klassendiagram mit Accounting

der SLAs zwischen zwei Providern gelistet, welche die gleichen Services anbieten. Für jede SLA zwischen Föderationspartner konstruiert der `CloudSimConfigurator` ein `SlaCsp`-Objekt, in welchem die Werte hinterlegt werden. Im Quellcode 4.1 in Abschnitt 4.5.1 XML-Erweiterungen wird eine beispielhafte Implementierung der XML-Struktur-Erweiterungen gezeigt, an der nachvollzogen werden kann, wie das Konzept der SLAs ausgebaut wird.

4.4.3 Scheduling-Eventfluss

Der Nachrichtenfluss des FCS zur Platzierung und Migration von VMs umfasst die initiale Konstruktion der VMs, entsprechend der auf Basis der Workloads erstellten Service-Requests und ihre Verteilung auf den zur Verfügung stehenden Hosts eines CSP. Sollte ein SLA-Manager feststellen, dass die Ressourcen eines Hosts nicht zur Bearbeitung der im zugewiesenen VMs ausreichen könnten, so wird per Nachricht an den Scheduler des Rechenzentrums, in dem der gefährdete Host ist, eine Migration von VMs eingeleitet. Ebenso abgedeckt wird der Nachrichtenfluss zur nächsthöheren Ebene. Sollte der Scheduler nicht in der Lage sein durch Migration innerhalb seines Rechenzentrums die Gefährdung des Hosts aufzulösen, so wird eine Anfrage zur Migration an den Broker auf CSP-Ebene gestellt. Dieser versucht nun die deallokierten VMs zu einem anderen angebundenen Rechenzentrum zu verschieben. Im Rahmen dieser Arbeit wird der Eventfluss ergänzt für die Ebene der Föderation. Es wird versucht VMs zu einem kooperierenden CSP zu migrieren, falls ihre Allokation in keinem eigenen Rechenzentrum gelingt. Die Abbildungen 4.7 und 4.8 veranschaulichen jeweils den Nachrichtenfluss einer Migration,

wobei erstere eine erfolgreiche, zweitere eine abgebrochene Migration dargestellt. Zu beachten ist, dass nur VMs aus eigenen Service-Requests anhand dieses Nachrichtenflusses migriert werden. Fremde VMs werden, falls nicht weiter unterstützt, zu ihren Besitzern zurückgesendet. Hierzu bedarf es keiner Anfrage, der Besitzer ist gezwungen diese anzunehmen und kann sie selbst neu distributieren. Eine Weiterleitung fremder VMs innerhalb einer Föderation wäre zwar potentiell möglich, da dies aber einen unerwünschten Kontrollverlust auf Seiten des ursprünglichen Dienstanbieters zur Folge hätte und ebenfalls unsinnige Abrechnungsschleifen entstehen könnten, wird auf diese Option verzichtet. Sobald von einem `FederatedDatacenterBroker` eine Migrations-Anfrage an einen `CloudBroker` erfolgt, wird von der `CloudBrokerStrategy` der CSP ausgewählt, welcher nach ihrer Taktik am meisten geeignet ist. Der `CloudBroker` versendet daraufhin eine `MIGRATION_REQUEST`-Nachricht an den `CloudBroker` des gewählten CSP. Dieser kann mit Hilfe seiner Strategie entscheiden, ob diese Anfrage angenommen wird. Im negativen Fall wird mit einer `MIGRATION_REQUEST_REJECT`-Nachricht geantwortet (nicht in den Abbildungen 4.7 und 4.8 dargestellt). Das Strategie-Objekt verwaltet zu den zu migrierenden VM eine Datenstruktur, in der ebenfalls die nach strategischen Gesichtspunkten sortierten CSPs gehalten werden, die noch nicht kontaktiert wurden. Der nächste, falls vorhandene Kooperationspartner würde folglich gefragt werden. Sollte keine Migration möglich sein, so werden die nicht migrierbaren VMs zurück an den `FederatedDatacenterBroker` in einer `SCHEDULE_REQUEST_CLOUDBROKER_REJ`-Nachricht gesendet. Im Falle einer positiven Antwort auf eine Migrationsanfrage kommt eine Migrationsvereinbarung zu Stande und die entsprechenden VMs sind an eine `MIGRATION_REMOTE`-Nachricht angehängt. Der empfangende `CloudBroker` leitet sie seinerseits zur Allokation an seinen `FederatedDatacenterBroker`. Der Ablauf einer erfolgreichen und einer abgebrochenen Migration sind bis zu dieser jeweils 7. Sequenz identisch. Abbildung 4.7 zeigt nun die Situation, dass im positiven Fall die migrierten VMs allokiert werden können und eine entsprechende Bestätigungsnachricht an den Ursprungsprovider versandt wird, während in Abbildung 4.8 der negative Fall abgebildet ist, dass der `FederatedDatacenterBroker`, anhand seiner Strategie auf CSP-Ebene, eine Allokation verweigert, die Strategie auf Föderations-Ebene überstimmt und durch eine `MIGRATION_REMOTE_REJECT`-Nachricht die Rückmigration veranlasst. Auf dem Ursprungsprovider werden diese VMs zur weiteren Behandlung dem eigenen `FederatedDatacenterBroker` übergeben. Es wird berücksichtigt, dass die Strategien auf den unterschiedlichen Ebenen konträre Ziele verfolgen. Für Broker und Scheduler werden daher unterschiedliche Verantwortungsbereiche festgelegt, die sich durch den Einflussbereich der Methoden ihrer Strategien ausdrücken. Die `AllocationPolicy` auf der untersten Ebene trifft für eine Migration, im Falle einer SLA-Gefährdung, die erste Auswahl an VMs und gibt ihre strategische Empfehlung eines Zielhosts ab. Der `FederatedDatacenterBroker` stellt die zentrale Scheduling-Instanz eines CSP dar. Da bei diesem alle Informationen eines CSP, über die Ressourcenauslastung aller Rechen-

zentren, über alle dort gehosteten VMs, fremde wie eigene, über alle migrierten, sowie abgewiesenen VMs und über die Wertigkeit aller damit verbundenen SLAs zusammenlaufen, erhält die `FederatedDatacenterBrokerStrategy` die Befugnis die Auswahl an VMs der `AllocationPolicy` zu überstimmen und durch eine eigene Auswahl zu ersetzen, solange dies ebenso zur Auflösung der Gefährdung führt. Es muss mindestens die gleiche Menge an Ressourcen freigegeben werden. Gleichfalls kann die Selektion allerdings auch beibehalten werden. In demselben Maße kann durch die Strategie bei der Wahl des Migrationsziels Einfluss genommen werden, indem entweder der Zielhost direkt administriert, oder ein Rechenzentrum bestimmt und dem dort lokalisierten Scheduler die Wahl des Host überlassen wird. Die `CloudBrokerStrategy` implementiert in erster Linie die Logik zur Wahl des passenden Kooperationspartners für eine Migration. Sie trifft aber keine Auswahl an VMs und ihre Entscheidung ein Migrations-Angebot anzunehmen, kann aufgehoben werden. Um Konflikte zu vermeiden bleibt die Möglichkeit innerhalb der Föderations-Strategie zur Entscheidung über ein Anfrage direkt auf die Methodik der `FederatedDatacenterBrokerStrategy` zurückzugreifen. Damit dies trotz möglicher, zeitlicher Verzögerungen zwischen den Abfragen (in Abbildung 4.7 in den Sequenzen 4 und 7) nicht zu unterschiedlichen Ergebnissen führt, wird die Möglichkeit hinzugefügt, Ressourcen zu reservieren. Gleichwohl bleibt die Möglichkeit, Strategien mit sich potentiell widersprüchlichen Entscheidungsprozessen zu testen. Es gibt vier verschiedene Szenarien, in denen ein VM an einen `FederatedDatacenterBroker` zurückgegeben wird und für alle wird das gleiche Ereignis mit der Markierung `SCHEDULE_REQUEST_CLOUDBROKER_REJ` verwendet:

1. Es wird kein CSP für eine Migrations-Anfrage gefunden.
2. Es werden ein oder mehrere Partner gefunden, jede Anfrage wird aber abgewiesen.
3. Es wurde ein Migrationspartner gefunden, aber nachdem die VM migriert wurde, wird sie nicht allokiert, sondern zurückgeschickt.
4. Eine VM wurde von einem kooperierendem Provider ausgeführt und wird zurückmigriert.

Während bei den letzteren Ausfallzeit bzw. Ausführungszeit vergangen ist, könnten die ersten beiden noch im gleichen Simulationszeitpunkt weiter verarbeitet werden. Daher unterscheidet der `FederatedDatacenterBroker` hierfür zwischen zwei Behandlungsroutinen. VMs aus den Szenarien 1. und 2. erhalten einen Simulationstick Ausfallzeit und es wird im Anschluss versucht sie neu zu allokieren. Im Fall der Szenarien 3. und 4. wird direkt probiert einen geeigneten Host zu finden. Sollte keine Platzierung in eigenen Rechenzentren gelingen so werden die VMs in allen Fällen erneut dem `CloudBroker` zur Distribution übergeben.

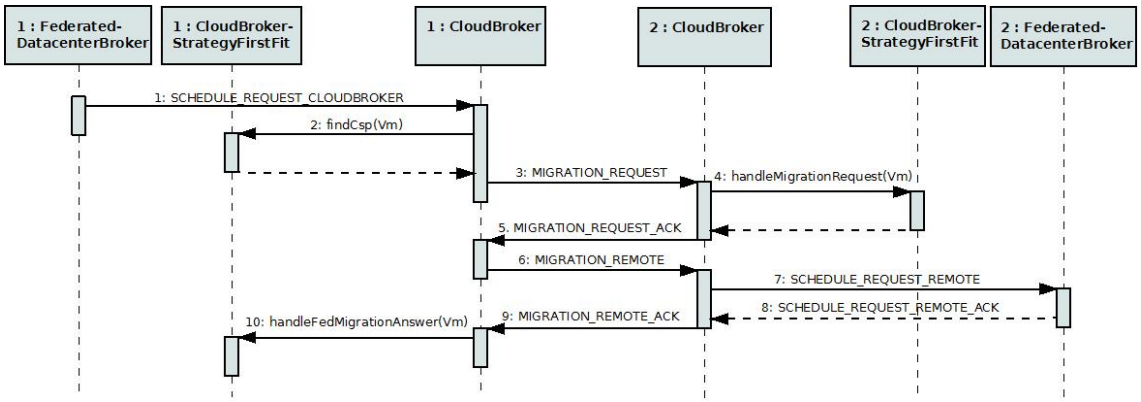

Abbildung 4.7: Nachrichtenfluss einer erfolgreichen Migrationen auf Föderations-Ebene. Es werden insbesondere die Methodenaufrufe und versendete Ereignisse (in Großbuchstaben) dargestellt.

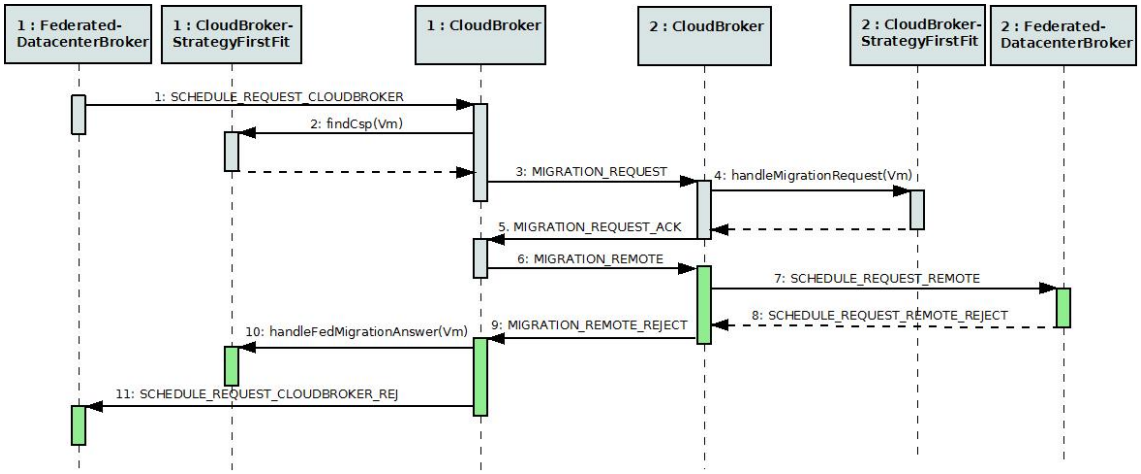

Abbildung 4.8: Nachrichtenfluss einer abgebrochenen Migrationen auf Föderations-Ebene. Es werden insbesondere die Methodenaufrufe und versendete Ereignisse (in Großbuchstaben) dargestellt.

4.4.4 Auktionsverfahren

Nachdem in Abschnitt 3.4.3 verschiedene Auktionsformen vorgestellt wurden und ihre Integration, sowie ihr Einfluss in Scheduling-Verfahren erläutert wurde, wird nun ein Modell vorgestellt, anhand dessen sich unterschiedliche Auktionsformen in FCS integrieren lassen. Anschließend wird die Struktur erklärt, welche den austauschbaren Auktionsstrategien zugrunde liegt, durch welche das taktische Verhalten der Auktionsteilnehmer gesteuert werden kann.

Auktionsmodell

Das Auktions-Modell erlaubt es verschiedenartige Auktionsverfahren in die Simulation von Cloud-Föderationen zu integrieren. Es ist auf einem Abstraktionsniveau angesiedelt, welches es ermöglicht sowohl unterschiedliche Auktionsformen nachzubilden, als auch das Auktionsobjekt zu variieren. Dazu werden die Schritte, die generell den Auktionsformen gemein sind erfasst. Ihr Zusammenwirken wird veranschaulicht und die Schnittstellen identifiziert, an denen strategische Entscheidungen der Teilnehmer den Verlauf beeinflussen. In Abbildung 4.9 ist veranschaulicht, wie das Modell in ein Scheduling-Verfahren auf Föderations-Ebene integriert ist. Scheduling im FCS ist auf dieser Ebene realisiert durch Anfragen eines Providers in die Föderation zur Übernahme einer VM. Dementsprechend wird im dargestellten Fall eine VM auktioniert und gemäß des Auktionsausgangs migriert. Auf Basis des Modells wurde ein Subsystem zur Durchführung von Auktionen entworfen, welches aufgrund der abstrakten Ebene, auf der das Modell angesiedelt ist, auch in beliebigen Verfahrensvarianten eingebunden werden kann. Der Ablauf gliedert sich grundsätzlich in die Aktivitäten:

1. *Starte Auktion*: Es werden die notwendigen Datenstrukturen konstruiert und initialisiert.

2. *Bestimme die Auktions-Teilnehmer*: Es werden in der Föderation Nachrichten zwischen den kooperierenden Providern ausgetauscht, um die Akteure zu ermitteln.

3. *Führe Auktionsrunde aus*: Dies ist der Hauptteil der Auktion. Es wird ein Angebot gemacht durch den Auktionator. Die vorher selektierten Teilnehmer können darauf bieten. Jede Auktion lässt sich in Runden einteilen, deren Anzahl aufgrund vorher definierter Regeln begrenzt ist. Dieser Schritt wird folglich iteriert, bis ein Abbruchkriterium erfüllt ist.

4. *Beende die Auktion*: Das Auktionsergebnis wird ermittelt und die Teilnehmer gegebenenfalls informiert.

An vier Positionen kann der Ablauf des Verfahrens durch die Teilnehmer beeinflusst werden. An diesen Stellen ist in Abbildung 4.9 die Unteraktivität `A_Strategy` dargestellt.

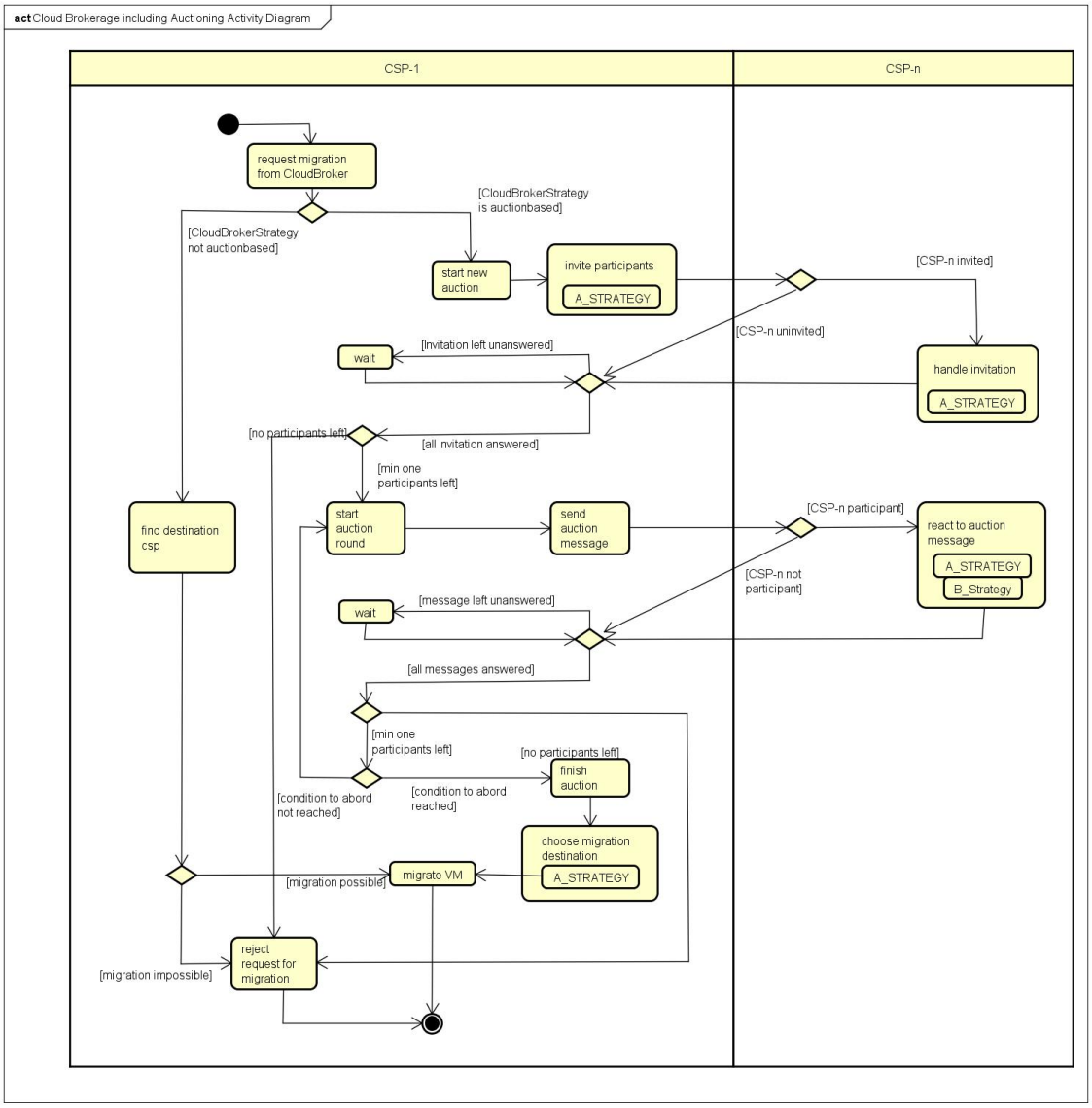

Abbildung 4.9: Modell des Ablaufs von Auktionsverfahren und ihre Integration in das Scheduling auf Föderations-Ebene

Der Auktionator kann anhand seiner Strategie entscheiden, welche CSPs zu der Auktion eingeladen werden, zudem hat er die Möglichkeit das Auktionsergebnis in einen komplexeren Auswahlprozess einzubinden, um ein Endresultat zu bestimmen. Auktionsteilnehmer können taktische Kriterien aufstellen, nach denen sie eine Einladung annehmen oder ausschlagen. Zur Durchführung einer Bietrunde müssen Gebote abgegeben werden. Zur Kalkulation der Gebote wird eine Biet-Strategie herangezogen, in Abbildung 4.9 mit `B_Strategy` gekennzeichnet, die variabel in der Auktionsstrategie gesetzt werden kann.

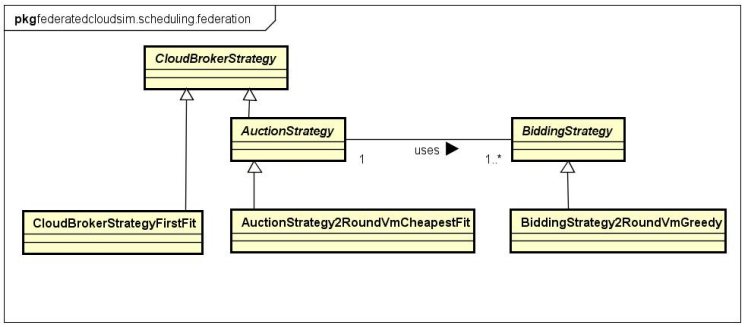

Abbildung 4.10: Vererbungshierarchie der Strategien auf Föderations-Ebene. Zur Verdeutlichung sind hier beispielhaft drei Strategien ausgewählt

Vererbungshierarchie der Auktionsstrategien

Abbildung 4.10 zeigt die Vererbungshierarchie, die genutzt wird, um die Anbindung des Auktionsverfahrens in das Scheduling auf Föderations-Ebene zu ermöglichen. Die abstrakte Klasse `AuctionStrategy` erbt dabei von der ebenfalls abstrakten Klasse `CloudBrokerStrategy`. Eine Auktionsstrategie erhält dadurch alle notwendigen Methoden und Attribute um den regulären Nachrichtenfluss zu unterstützen und kann daher im Simulationsverlauf die Strategie eines `CloudBroker` ersetzen. Sie erweitert ihre Eigenschaften allerdings um weitere abstrakte Funktionen, welche die strategischen Entscheidungen in dem zuvor vorgestellten Auktions-Modell realisieren. Insbesondere besitzt eine Auktions-Strategie eine Biet-Strategie, welche die Logik realisiert wie ein Auktionsgebot berechnet wird. Auch in diesem Fall muss eine konkrete Instanz von einer abstrakten Basisklasse `BiddingStrategy` abgeleitet sein.

In Abschnitt 4.4.5 werden die Scheduling-Verfahren auf Auktionsbasis vorgestellt. Das Vorgehen der Biet-Strategien zur Berechnung von Auktionsgeboten kann durch die Implementierungs-Beschreibung in Abschnitt 4.5.3 nachvollzogen werden, in dem auch die für diese Arbeit implementierten Klassen des Auktionssystems detailiert erklärt sind.

4.4.5 Scheduling-Verfahren

Scheduling wird im FCS notwendig, wenn auf Rechenzentrums-Ebene ein Host gefährdet ist. Der SLA-Manager stellt fest, dass die Ressourcen dieses Hosts nicht oder potentiell nicht ausreichend sind, um den Bedarf der auf ihm allokierten VMs zu decken. Zur Beurteilung sind zwei Grenzwerte konfigurierbar. Liegt die Auslastung eines Host über der Gefährdungsgrenze, so wird unmittelbar ein Rescheduling initiiert. Dies geschieht ebenso, falls ein Host in aufeinanderfolgenden Zeiträumen über der Beobachtungsgrenze liegt. Nachdem Pasternak (2015) [33] eine Vielzahl an Verfahren implementiert hat, die Migrationen auf der Rechenzentrums-Ebene steuern, liegt der Fokus dieser Arbeit auf Scheduling-Verfahren auf CSP-Ebene und Föderations-Ebene. Prioritäres Ziel aller Strategien ist es,

die Wirtschaftlichkeit des Providers zu erhöhen. Dazu wird versucht einzelne oder mehrere Faktoren des genutzten Finanz-Modells zu optimieren. Die entworfenen Strategien des CSP-Brokers unterscheiden sich durch ihre Kombination von Methoden zur VM-Auswahl und Host-Auswahl. Den Brokern auf diesem Level stehen alle VMs auf allen Hosts aller Rechenzentren als Migrationsobjekte, sowie alle Hosts aller angeschlossenen Rechenzentren als Migrationsziele zur Verfügung. Die einzige Bedingung, die bei der Selektion erfüllt werden muss, ist, dass die Gefährdung des initial bedrohten Hosts aufgelöst wird. Es wurden für jede VM-Auswahl zwei Alternativen entworfen. In der ersten Varianten werden lediglich die VMs des gefährdeten Host betrachtet und die Auswahl des Schedulers unter Umständen verändert. In der zweiten Variante werden alle zur Verfügung stehenden VMs betrachtet, eine globale Auswahl getroffen und zusätzliche Migrationen durch eine potentielle Umverteilung in Kauf genommen. Auf Föderations-Ebene liegt die Hauptaufgabe der Verfahren in der Wahl des CSP, welcher die Ausführung einer VM kooperativ übernehmen soll. Eine VM-Auswahl findet nicht statt, wodurch die Auswahl-Strategien auf CSP-Ebene für sie ebenso Relevanz haben für den Fall, dass dort keine Migration möglich ist. Nachfolgend sind verschiedene Algorithmen und Kombinationen von ihnen beschrieben, welche diese Wahlen ausführen.

VM-Auswahl nach geringstem Ressourcen-Verbrauch

Diese Strategie berücksichtigt den aktuellen Ressourcen-Verbrauch einer VM und es werden diejenigen VMs gewählt, welche den geringsten Verbrauch aufweisen. Es werden mehrere Ziele verfolgt, kleinere VMs sollten einfacher in eigenen Rechenzentren zu platzieren sein und damit erstens durch ihre Migration eine bessere Balance der Auslastung herbeiführen und zweitens die Migrationen in die Föderation verringern. Weiterhin würde eine Migration von ihnen in die Föderation keinen hohen Verlust auf der Einnahmenseite eines Providers bedeuten.

VM-Auswahl nach höchstem Ressourcen-Verbrauch

Genau wie die zuvor beschriebene Strategie werden die VMs nach aktuellem Ressourcenverbrauch bewertet. In dieser Abwandlung werden nun die größten VMs migriert. Die Anzahl der Migrationen könnte sich deswegen reduzieren, wodurch weniger Strafzahlungen entrichtet werden müssten. Da die Quantität der Migrationen reduziert werden soll und sich diese bei einer Umverteilung aller Rechenzentren vergrößert, wird diese Strategie nicht in zwei Varianten realisiert, sondern nur die VMs des gefährdeten Host betrachtet.

VM-Auswahl nach geringstem SLA-Level

Dieser Algorithmus migriert vorzugsweise VMs, welche im Rahmen eines SLA ausgeführt werden, der ein niedriges Level aufweist. Sie können daher nicht nur öfter migriert werden

ohne Strafzahlungen zu befürchten als höherwertige VMs, sondern falls eine Strafe entrichtet werden muss, fällt diese auch vergleichsweise geringer aus. Hinzu kommt das eine Migration in die Föderation einen vergleichsweise kleineren Verlust potentieller Einnahmen darstellt.

VM-Auswahl nach geringster SLA-Gefährdung

Strafzahlungen werden erst fällig, wenn die Ausfallzeit einer VM den in dem zugehörigen SLA festgelegten Grenzwert übersteigt. Anhand dieses Grenzwerts wird für jede VM ihre prozentuale Gefährdung errechnet und es werden die am wenigsten gefährdeten bei der Auswahl priorisiert. Das Ziel ist es den Gefährdungsspielraum besser auszunutzen und so Strafzahlungen zu reduzieren.

VM-Auswahl nach höchster SLA-Gefährdung

In dieser Strategie werden VMs gewählt, die möglichst nah an ihrem Grenzwert liegen. Für besonders ausgelastete CSPs kann es von Vorteil, sein gefährdete VMs in die Föderation abzugeben. Falls dies möglich ist, so kann es sein, dass der Kooperationspartner eher in der Lage ist den Ressourcenbedarf zu decken und Strafzahlungen würden so vermieden werden. Versagt der Föderationspartner, so muss er zumindest einen Teil der Strafzahlung mittragen. VMs, deren Gefährdungsniveau über 90 Prozent liegt, werden nicht berücksichtigt, um zu verhindern, dass diese migriert werden, obwohl noch Alternativen vorhanden sind.

Host-Auswahl FirstFit

Der erste passende Host wird gewählt.

Host-Auswahl BestFit

Es wird der Host gewählt, welcher die geringste prozentuale Auslastung hat. Wenn die Auslastung der Host auf ein ähnliches Level gebracht werden kann, so könnte sich die Anzahl der nötigen Migrationen verringern.

Host-Auswahl LowestDanger

Es wird für jede VM die prozentuale SLA-Gefährdung und darauf basierend für jeden Host die durchschnittliche Gefährdung errechnet. Es wird der Host mit der geringsten, durchschnittlichen, prozentualen SLA-Gefährdung bevorzugt. Diese Hosts konnten in der Vergangenheit ihre Ressourcen gut managen. Es wird sich durch diese Auswahl erhofft, dass dies auch mit einer zusätzlichen Belastung der Fall ist. Zudem würden, falls doch eine Überlastung auftritt, vergleichsweise gering gefährdete VMs von diesem Host wegmigriert.

Insgesamt sollten die entstehenden Strafzahlungen aus Ausfallzeiten reduziert werden können.

CSP-Auswahl FirstFit

Der erste kooperierende CSP wird selektiert.

CSP-Auswahl FairFit

Nach dem Rundlauf-Verfahren werden alle CSPs der Föderation gleich oft für eine Migration angefragt.

CSP-Auswahl CheapestFit

Es wird der bezüglich bestehender SLAs günstigste Anbieter gewählt und damit die Kosten für die Auslagerung von VMs optimiert.

CSP-Auswahl RelianceScope

Es wird der CSP priorisiert, welcher den besten Vertrauenswert aufweist und damit am unwahrscheinlichsten eine Strafe produziert (siehe Abschnitt 3.4.5).

CSP-Auswahl durch Kombinierte Verfahren

Ein guter Scheduler implementiert einen geeigneten Kompromiss oder verwendet eine Kombination von Algorithmen in Abhängigkeit des Einsatzgebiets (vgl. Teng 2011 [40], S.28). Folglich wird versucht, Verfahren zu entwerfen, welche mehrere wirtschaftliche Einflussgrößen optimieren. Die Auswertung des Vertrauensfaktor wird mit der Bewertung anhand der finanziell günstigsten Vereinbarung auf zwei verschiedene Arten kombiniert. Im Algorithmus *CheapestReliability* werden die CSPs zunächst nach ihrer Vertrauensfaktor-Bewertung sortiert. Im Anschluss wird ein konfigurierbarer, prozentualer Anteil an CSPs aussortiert. Die CSPs mit der schlechtesten Bewertung können folglich nicht gewählt werden, wenn aus den Verbleibenden der Günstigste selektiert wird. Das Verfahren *MostReliableQuotation* geht analog vor, nur wird zuerst nach der finanziellen Bewertung sortiert, anschließend ein Anteil eliminiert und aus den Verbleibenden der bezüglich des Vertrauensfaktor Zuverlässigste ausgewählt.

Auktionsbasierte CSP-Auswahl CheapestFit

Es wird ein Auktions-Verfahren durchgeführt und die teilnehmenden Föderationspartner haben die Möglichkeit den Ausführungspreis für eine VM zu gestalten. Am Ende dieser dynamischen Preisfindung wird der günstigste Anbieter gewählt. Ziel ist es die Auslagerungskosten zu reduzieren und Auktionseffekte zu nutzen.

Auktionsbasierte CSP-Auswahl CheapestReliability

Eine prozentuale Teilmenge der föderierten CSPs, welche die schlechtesten Vertrauensfaktor-Werte hat, wird nicht zu einer Auktion eingeladen. Der am Ende der Auktion günstigste Provider erhält die zu migrierende VM. Der Prozentsatz ist der Strategie ist über die XML-Konfiguration einstellbar.

Auktionsbasierte CSP-Auswahl MostReliableQuotation

Bei dieser Strategie wird eine Auktion veranstaltet. Nach der Beendigung wird ein prozentualer Anteil der teuersten CSPs aussortiert. Von den verbliebenen CSPs wird derjenige bevorzugt, welcher den höchsten Vertrauensfaktor-Wert hat.

4.5 Implementierung

In diesem Abschnitt werden zu den in Unterkapitel 4.4 beschriebenen Konzepten konkrete Implementierungen und ein Überblick über den entwickelten Code gezeigt. In Abschnitt 4.5.1 wird die Erweiterung der XML-Dateien erörtert. Einen Überblick über die Klassen des FCS, die erweitert wurden, gibt Abschnitt 4.5.2. Die für diese Arbeit insbesondere relevanten Anpassungen der Scheduling-Komponenten und neu hinzugefügten Ergänzungen werden mit ihrer Struktur in Abschnitt 4.5.3 beschrieben.

4.5.1 XML-Erweiterungen

Damit anhand eines Finanz-Modells berechnet werden kann, welche Einnahmen und Ausgaben anfallen, sind Preise für die zur Kalkulation herangezogenen Abrechnungs-Einheiten und potentielle Einflussfaktoren festzulegen. Diese Steuerungsgrößen müssen namentlich in der XSD-Datei bestimmt werden. Dazu wird die XSD-Datei an zwei Stellen erweitert. Die Struktur-Definitionen der komplexen Typen `csp` und `cspSla` werden jeweils um ein identisches Element `accountingFactors` ergänzt. Diese Elemente sind selbst komplexe Typen, deren Kind-Elemente Gruppen von Kalkulations-Parameter darstellen. Angepasst an das gewählte Finanz-Modell können in einer oder mehreren Gruppen die Steuerungsgrößen aufgelistet werden. Der Quellcode 4.1 zeigt die Kennzeichnung der Parameter im `csp-Typ` gemäß des Finanz-Modells aus Abschnitt 3.4.4. Die Kennzeichnung im `cspSla-Typ` erfolgt analog. In der XML-Konfigurationsdatei hat ein Nutzer des Simulations-Frameworks infolgedessen die Möglichkeit für jeden CSP den Parametern Werte zuzuordnen, die zusammen in die Gestaltung der zwischen dem Provider und seinen Kunden ausgehandelten SLAs einfließen. In der XML-Konfiguration der SLAs zwischen CSPs können denselben Parametern andere Werte zugewiesen werden, welche damit vertraglich geschlossene Vereinbarungen darstellen, mithilfe derer die Abrechnung kooperativ erbrachter Services erfolgt.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema id="FederatedCloudConfig"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  ...
 <xs:complexType name="cspType">
 <xs:sequence>
  ...
  <xs:element name="accountingFactors" maxOccurs="1" minOccurs="0">
   <xs:complexType>
    <xs:sequence>
     <xs:element name="ressourceFactors" maxOccurs="1" minOccurs="1">
      <xs:complexType>
       <xs:attribute name="cpuCostFactor" type="xs:string" use="required" />
       <xs:attribute name="ramCostFactor" type="xs:string" use="required" />
      </xs:complexType>
     </xs:element>
     <xs:element name="slaFactors" maxOccurs="1" minOccurs="1">
      <xs:complexType>
       <xs:attribute name="resourceSlaFactorGold" type="xs:string" use="required" />
       <xs:attribute name="resourceSlaFactorSilver" type="xs:string" use="required" />
       <xs:attribute name="resourceSlaFactorBronze" type="xs:string" use="required" />
      </xs:complexType>
     </xs:element>
     <xs:element name="penaltyFactors" maxOccurs="1" minOccurs="1">
      <xs:complexType>
       <xs:attribute name="downtimeFactor" type="xs:string" use="required" />
       <xs:attribute name="brokenCpuFactor" type="xs:string" use="required" />
       <xs:attribute name="brokenRamFactor" type="xs:string" use="required" />
       <xs:attribute name="initialDowntimePenaltyGold" type="xs:string" use="required" />
       <xs:attribute name="initialDowntimePenaltySilver" type="xs:string" use="required" />
       <xs:attribute name="initialDowntimePenaltyBronze" type="xs:string" use="required" />
       <xs:attribute name="continuingDowntimePenaltyGold" type="xs:string" use="required" />
       <xs:attribute name="continuingDowntimePenaltySilver" type="xs:string" use="required" />
       <xs:attribute name="continuingDowntimePenaltyBronze" type="xs:string" use="required" />
      </xs:complexType>
     </xs:element>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
  ...
 </xs:sequence>
 <xs:attribute name="id" type="xs:string" default="cspId0" />
 <xs:attribute name="name" type="xs:string" use="required" />
 <xs:attribute name="cloudBrokerStrategy"
   type="xs:string" use="optional" />
 <xs:attribute name="dcBrokerStrategy"
   type="xs:string" use="optional" />
 <xs:attribute name="slaManagerStrategy"
   type="xs:string" use="optional" />
</xs:complexType>
```

Quellcode 4.1: Ausschnitt aus der Definition des Schemas eines CSPs

```
1 public boolean configureCloudSim() {
2 ...
3 if (cloudConfig.getRootElement().getChild("cspSlas") != null)
4  for (Element cspSlas : cloudConfig.getRootElement().getChild("cspSlas").getChildren("cspSla")) {
5   SlaCsp sla = new SlaCsp(getCspMap().get(cspSlas.getAttributeValue("providingCspId")),
6         getServiceMap().get(cspSlas.getAttributeValue("serviceId")),
7         getCspMap().get(cspSlas.getAttributeValue("utilizingCspId")));
8   getCspServiceMap().get(cspSlas.getAttributeValue("utilizingCspId")).
9         add(cspSlas.getAttributeValue("serviceId"));
10  if (cspSlas.getChild("accountingFactors") != null) {
11   for (Element elem : cspSlas.getChild("accountingFactors").getChildren()) {
12    for (Attribute attribute : elem.getAttributes()) {
13     String name = attribute.getName();
14     String value = attribute.getValue();
15     sla.getAccountingFactors().put(name, value);
16    }
17   }
18  }
19 }
20 ...
21 }
```

Quellcode 4.2: Ausschnitt aus der Methode `configureCloudSim()` zum Auslesen der SLA-Parameter aus der XML-Datei

Im Quellcode 4.2 ist ein Ausschnitt aus der Methode `configureCloudSim()` gezeigt, die bei der Initialisierung der Simulationsumgebung in der Klasse `CloudSimConfigurator` ausgeführt wird. Zu beachten ist, dass in einer for-Schleife alle `cspSla`-Elemente durchlaufen werden und für jedes ein `SlaCsp`-Objekt erzeugt wird. Falls Kalkulations-Parameter gesetzt sind, werden diese in eine String-Double-Zuordnung des Sla-Objekts gespeichert. Auf diese Weise können für verschiedenste Modelle beliebige Steuerungsgrößen inklusive ihrer Wertbelegung eingelesen werden. Da das sich die Kunden-SLAs nur bezüglich des SLA-Levels unterscheiden, werden diese Werte der Steuerungsgrößen der Einfachheit halber mit derselben Methodik in eine Zuordnungsstruktur der Kalkulations-Strategie eingelesen, anstatt in jedem SLA-Objekt gespeichert zu werden.

4.5.2 Übersicht der Quellcode-Erweiterungen

Es werden die Java-Klassen des FCS vorgestellt, welche in dieser Arbeit angepasst oder erweitert wurden. Für eine umfassende Beschreibung der Klassen des FCS sei verwiesen auf Spohr (2015) [38]. Die Paketstruktur wurde nur durch Hinzunahme des *account*-Pakets verändert.

- *com.materna.federatedcloudsim*
  - `CloudServiceProvider` wird um eine Abrechnungs-Komponente, sowie einen Auktionsmanager erweitert.

- `FederatedVm` erhält zusätzlich verschiedene Attribute, um Entgelte und Strafen anhand des Status, der Nutzungsausfallzeit und der Migrationshistorie berechnen zu können.

- *com.materna.federatedcloudsim.account*

  - `Accounting` wird für jeden CSP instanziiert und ermöglicht die Abrechnung nach einem Finanz-Modell.
  - `AccountingStrategy` ist die abstrakte Klasse von der konkrete Strategien zur Abrechnung abgeleitet werden müssen.
  - `AccountingStrategyBasic` ist die konkrete Implementierung des in Abschnitt 3.4.4 vorgestellten Finanz-Modells.

  Dieses neu hinzugefügte Paket dient der Umsetzung des in Abschnitt 3.4.4 vorgestellten Finanz-Modells anhand des in Abschnitt 4.4.1 beschriebenen Konzepts.

- *com.materna.federatedcloudsim.core*

  - `FederatedCloudSimTags` wird um Tags zur Markierung von Nachrichten erweitert, für das Scheduling auf Föderationsebene, daraus resultierende Nachrichten an den `FederatedDatacenterBroker` und Nachrichten zur Durchführung von Auktionen.

- *com.materna.federatedcloudsim.federation.sla*

  - `SlaCsp` erhält eine zusätzliche Datenstruktur zur Speicherung der finanziellen Parameter.
  - `SlaManager` wird um Methoden zur Verwaltung der SLAs mit finanziellen Parametern ergänzt.

- *com.materna.federatedcloudsim.monitoring*

  - `FileLog` erhält eine Methode zur Ausgabe einer neuen Protokolldatei, welche verschiedene Metriken zur Finanz-Kalkulation der einzelnen CSPs und der Föderation zusammenträgt.

- *com.materna.federatedcloudsim.scheduling.csp*

  - `FederatedDatacenterBroker` erhält zusätzliche Attribute und Methoden zur Verwaltung von VMs, welche bei der Migration auf Föderations-Ebene abgewiesen wurden.
  - `FederatedDatacenterBrokerStrategyABC` Das Framework wird mit den in Abschnitt 4.4.5 beschriebenen Strategien zur Migration auf CSP-Ebene ergänzt.

- *com.materna.federatedcloudsim.scheduling.federation*

  - `Auction` wird für jede Auktion instanziiert.
  - `AuctionManager` wird für jeden CSP instanziiert und verwaltet alle Auktionen.
  - `AuctionMessage` wird zu jede Kommunikationsbeziehung zweier Provider erzeugt und verwaltet die ausgetauschten Informationen eines Bietvorgangs.
  - `AuctionStrategy` ist eine abstrakte Klasse, von der konkrete Strategien, welche beim auktionsbasierten Scheduling die Entscheidungslogik eines Providers implementieren, abgeleitet werden müssen.
  - `BiddingStrategy` stellt die abstrakte Klasse dar, von der Biet-Strategien, welche die Logik zur Berechnung eines Auktionsgebots realisieren, erben müssen.
  - `CloudBroker` wird um Attribute und Methoden erweitert zur Durchführung von Scheduling-Verfahren auf Föderations-Ebene und zur Verwaltung mirgrierter VMs.
  - `CloudBrokerStrategy` ist die abstrakte Klasse von der konkrete Strategien zur Migration von VMs in einer Cloud-Föderation abgeleitet werden müssen. In Abschnitt 4.5.3 wird diese Klasse eingehend beschrieben.
  - `CloudBrokerStrategyABC` Entsprechend der in Abschnitt 4.4.5 beschriebenen Verfahren wird der Simulator um Strategien zur Migration auf Föderations-Ebene ergänzt.

  In Abschnitt 4.5.3 werden ebenfalls die einzelnen Klassen zur Durchführung von Auktionen und ihr Zusammenwirken detailliert erörtert.

4.5.3 Realisierung der Scheduling-Komponenten

In diesem Unterkapitel wird zunächst die Realisierung der abstrakten Klasse von Scheduling-Strategien auf Föderations-Ebene näher erläutert. Anschließend werden die Struktur der Auktions-Komponenten, ihre Schnittstellen zu den Scheduling-Klassen und die Implementierung der Biet-Strategien beschrieben.

Scheduling auf Föderations-Ebene

Alle Scheduling-Strategien auf Föderations-Ebene, die schon implementiert sind müssen und jede, die von einem Nutzer des Simulator neu entwickelt wird muss von der Basisklasse `CloudBrokerStrategy` erben. Diese abstrakte Klasse gibt durch eine Reihe von Methoden vor, welche Funktionalität eine konkrete Strategie mindestens aufweisen muss, um in den Programmfluss eingefügt werden zu können. Diese Methoden werden hier vorgestellt und ihre Funktion in einem Scheduling-Verfahren erklärt, anhand von Abbildung 4.7 kann ihre Position im Nachrichtenfluss einer Migration nachvollzogen werden.

`findCsp(FederatedVm)` liefert anhand strategischer Überlegungen einen kooperierenden CSP, an den eine Migrations-Anfrage gestellt werden kann.

`handleMigrationRequest(FederatedVm)` beurteilt, ob eine Migrations-Anfrage angenommen oder abgelehnt werden sollte.

`handleRejectedMigrationRequest(FederatedVm)` nutzt eine Datenstruktur, die schon kontaktierte CSPs verwaltet und implementiert die Logik, im Falle einer Migrations-Absage, den nächsten, sinnvollen Kooperationspartner zu finden.

`handleFedMigrationAnswer(FederatedVm)` ist dafür zuständig, nach Abschluss von erfolgreichen und auch fehlgeschlagenen Migrationen, alle genutzten Datenstrukturen zu setzen bzw. zurück zu setzen.

`isAuctionBased()` liefert die Art der Strategie, damit der `CloudBroker` entweder eine Auktion oder eine normale Migration initialisiert.

In Abschnitt 4.4.5 wurden die Scheduling-Verfahren erläutert, um die der Simulator erweitert wurde.

Auktionsbasiertes Scheduling

Zur Durchführung einer Auktion kommt es, falls der `FederatedDatacenterBroker` eine VM nicht innerhalb der Rechenzentren seines CSPs mit notwendigen Ressourcen versorgen kann, daher eine Migration auf Föderations-Ebene anstößt und die Strategie des `CloudBrokers` auktionsbasiert ist. Die für die Auktionierung benötigten Klassen sind in Abbildung 4.11 dargestellt. Falls eine VM migriert werden soll, informiert der `CloudBroker` den `AuctionManager` des CSP und dieser startet eine Auktion indem er zu der VM ein `Auction`-Objekt erzeugt, seine Datenstrukturen zur Verwaltung aktualisiert und von der Auktions-Strategie die Teilnehmer bestimmen lässt. Der `CloudBroker` verschickt an alle Teilnehmer eine Einladungsnachricht, an der eine `Auction` angehängt ist. Jede `Auction` erhält bei ihrer Konstruktion die Auktionsform der Strategie und ein Objekt, das auktioniert werden soll. In der vorliegenden Implementierung ist die Auktionsform ein VM-Versteigerung, für die zwei Bietrunden durchgeführt werden und das Objekt folglich eine VM. Die Auktionsform wird durch einen String repräsentiert, der durch potentielle Teilnehmer getestet wird, so dass nur CSPs an einer Auktion teilnehmen, welche die gleiche Logik einer Versteigerungsform umsetzen. Die Implementierung eines allgemeinen Auktionsobjekts ermöglicht die Umsetzung andersartiger Auktionen, bei denen keine VM versteigert wird. Falls eine kompatible Strategie vorliegt kann ein eingeladener CloudBroker entscheiden, ob eine Teilnahme erwünscht ist. Hier könnte in Abhängigkeit der Auslastung der Rechenzentren entschieden werden, die in Abschnitt 4.4.5 vorgestellten Strategien testen, ob es generell einen Host gibt, der die VM aufnehmen könnte. Nachdem die Rückmeldung

aller eingeladenen Teilnehmer erfolgt ist, erhält jeder Bieter eine `AuctionMessage`. Jedes Nachrichtenobjekt kennt die zugehörige Auktion, den Bieter, die Runde der Versteigerung und ein Bieter kann in einer Zuordnung, wie sie schon für die Kalkulations-Parameter genutzt wurde, seine Werte eintragen, welche ein bindendes Gebot repräsentieren. Dabei wird ein Ausführungspreis geboten, der sich aus den Werten zusammensetzt, die zur Berechnung der Ressourcen-Nutzung herangezogen werden. Im Rahmen der durchgeführten Auktionen wird auf die Anpassung der Strafzahlungs-Parameter verzichtet und auf die per SLA bestehende Vereinbarung zurückgegriffen. Eine `AuctionMessage` kann alternativ zurückgewiesen werden. Zur Berechnung eines Gebotes besitzt jede Auktions-Strategie eine Biet-Strategie, welche im sich anschließenden Absatz genauer beschrieben werden. Sind alle Nachrichten wieder beim Auktionator eingetroffen, so werden die Gebote ausgewertet und eine zweite Runde gestartet, wobei der Bieter mit dem günstigsten Gebot hiervon informiert wird. Allerdings nutzt keine der bisher implementierten Strategien diese Information. Im Anschluss an eine zweite und finale Bietrunde wird der Gewinner durch die Auktions-Strategie identifiziert, falls Bieter übrig geblieben sind und die VM migriert. Es wird dazu der in Abschnitt 4.4.3 beschriebene Nachrichtenfluss genutzt. Aufgrund der abgeschlossenen Auktion mit einem bindenden Gebot erhält der `CloudBroker` des Bieters keine Möglichkeit zur Abweisung. Dem `FederatedDatacenterBroker` bleibt diese Alternative bestehen. Es wird die Gebührenvereinbarungen der SLA zwischen den nun kooperierenden Providern nicht verändert, da sich die Entgelte aus dem Auktionsgebot nur auf die zugehörige VM und auch nur temporär beziehen. Stattdessen wird die Berechnung des Zahlungsausgleichs durch die Abrechnungskomponente des Bieters übernommen. Eine VM kann dazu als auktioniert markiert und entsprechend verrechnet werden. Falls eine Auktion nicht erfolgreich beendet werden kann, wird die Migration dem `FederatedDatacenterBroker` als abgewiesen gemeldet.

Bietstrategien

Die Bietstrategie ist dafür zuständig ein verbindliches Gebot zu berechnen, um dem Auktionator einen Ausführungspreis für eine VM anbieten zu können. In der ersten Runde greifen alle implementierten Strategien auf die Parameterwerte ihrer Kunden-SLA zurück und nehmen erst in der zweiten Runde eine prozentuale Preisanpassung vor, welche sich folglich auf den üblicherweise einem Kunden in Rechnung gestellten Preis bezieht. Eine Ausnahme stellt dabei die `BiddingStrategy2RoundVmDeluxe`-Variante dar, da sie eine Preiserhöhung versucht durchzusetzen.

`BiddingStrategy2RoundVmCautious`

: Diese vorsichtige Strategie gewährt in der finalen Runde einen Nachlass von fünf Prozent.

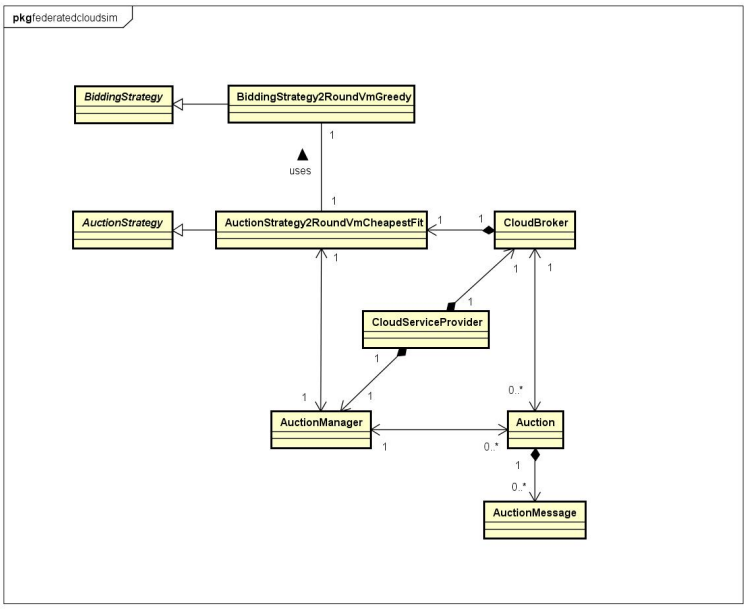

Abbildung 4.11: Struktur der Auktions-Komponenten und ihre Anbindung an die Scheduling-Struktur des FCS. Beispielhaft sind zwei konkrete Strategie-Implementierungen gewählt.

`BiddingStrategy2RoundVmStandard`

Die Standard-Variante verringert den Ausführungspreis in der zweiten Runde um 20 Prozent.

`BiddingStrategy2RoundVmGreedy`

Um 70 Prozent werden die Parameter-Werte beim letzten Gebot gesenkt.

`BiddingStrategy2RoundVmDeluxe`

Diese Luxus-Strategie verdoppelt den Preis, der verlangt wird, im Vergleich mit einem Kunden-SLA schon in der ersten Runde und behält diesen Kurs auch final.

`BiddingStrategy2RoundVmVariable`

Diese Strategie verfolgt einen komplexeren Ansatz als die Übrigen. Zunächst wird der durchschnittliche, prozentuale, aktuelle Nutzungsgrad der Rechenzentren des Providers errechnet. Dieser Wert wird mit verschiedenen Grenzen verglichen und abhängig von dem Vergleichsergebnis wird eine der vier oben genannten Strategien umgesetzt:

- Sollte die Auslastung durchschnittlich über der vom SLA-Manager vorgesehenen Gefährdungsgrenze liegen, so wird die Luxus-Variante gewählt.
- Liegt der Wert darunter, aber über der Beobachtungsgrenze berechnet die vorsichtige Strategie den Angebotspreis.
- Ist der Nutzungsgrad geringer, jedoch über der anhand des Finanz-Modells berechneten prozentualen Auslastung, die nötig wäre, um die laufenden Kosten zu decken, so wird ein Preisnachlass wie in der Standard-Strategie gewährt.

- Falls die Auslastung unter diesem Wert liegt, so wird die extremste Variante mit 70 Prozent Rabatt gewählt.

4.5.4 Weitere Anpassungen

VM-Rückholung

Es wurde die Möglichkeit implementiert, VMs von einem Föderationspartner zurückzufordern. Dazu kann eine CloudSim-Nachricht versandt werden, wordurch die VM deallokiert, zurück migriert und neu allokiert wird. Dem Kooperationspartner wurde dabei kein Vetorecht eingeräumt. Keine der implementierten Strategien nutzt diese Variante, eine proaktive Konsolidierung, abhängig von dem Vertrauensfaktor, wäre eine interessante Überlegung.

Kapitel 5

Simulation und Evaluierung

In diesem Kapitel wird beschrieben unter welchen Bedingungen Simulationen durchgeführt wurden, um die vorgestellten Scheduling-Verfahren zu testen. Dazu werden als erstes die verwendeten Workloads ananlysiert. Im Anschluss wird die Konfiguration des Simulations-Frameworks beschrieben. Es folgen verschiedene Simulationsphasen in denen unterschiedliche Szenarien durchgespielt werden, um einzelne Effekte besser zu extrahieren. Nach jeder Phase werden die Ergebnisse zusammengefasst und ihre Implikationen für die nächste Phase beschrieben. Das Kapitel schließt mit einer Zusammenfassung der Ergebnisse.

5.1 Analyse der Workloads

Zur Simulation stehen zwei Datensätze zur Verfügung. Der erste ist ein frei zugänglicher Workload, der von dem Unternehmen Solvinity [1] in Kooperation mit der TU-Delft 2014 im Grid Workload Archiv veröffentlicht wurde (siehe [2]). Er enthält die Leistungsmetriken, die in verteilten Rechenzentren bei der Verarbeitung von geschäftskritischen Anwendungen erfasst wurden. Dies waren vor allem Applikationsserver wie z.B. Datenbank- und Werbserver. Er wurde zur Analyse unternehmenskritischer Cloud-Umgebungen der Öffentlichkeit zur Verfügung gestellt. Es sind Traces über vier Monate enthalten, drei mit jeweils 500 VMs und ein Monat mit 1250 VMs. Der zweite Datensatz wurde von der Firma Materna GmbH [2] zur Verfügung gestellt und enthält Traces über drei Monate, die ebenfalls bei der Verarbeitung unternehmenskritischer Anwendungen aufgezeichnet wurden. Es sind 520, 527 und 547 VMs erfasst und der Workload wird in Kürze der Forschungsgemeinde zur Verfügung gestellt.

[1] zu finden unter `https://www.solvinity.com/`
[2] zu finden unter `http://www.materna.de/`

5.2 Konfiguration der Simulationsumgebung

Das Simulations-Framework bietet umfangreiche Möglichkeiten zur Konfiguration von Cloud-Infrastrukturen[3]. Die Datensätze enthalten keine Angaben zu der genutzten Infrastruktur. Zudem muss davon ausgegangen werden, dass sie für derartige geschäftskritische Anwendungen mehrfach überdimensioniert und damit für Tests von Scheduling-Verfahren eher ungeeignet war. Es wird im Folgenden beschrieben wie eine Konfiguration ausgewählt und für alle weiteren Simulationen beibehalten wurde, um den Fokus auf den Vergleich der Scheduling-Strategien zu lenken. Es wurden fünf der sieben Monatstraces ausgewählt und jeweils einem CSP zugewiesen, so dass eine Föderation mit fünf Partnern entsteht, für die eine Simulation von einem Monat durchgeführt werden kann. Aufgrund der eher geringen Anzahl an VMs pro Workload haben dabei alle Provider ein Rechenzentrum mit einer Ausnahme. Für die Bearbeitung von 1250 VMs ist ein Provider mit zwei Rechenzentren ausgestattet. Die eingesetzten Hosts wurden entsprechend marktüblicher Server-Hardware ausgewählt mit 30 CPU-Kernen@2,9GHz und 50GB Arbeitsspeicher. Im nächsten Schritt werden die Parameter gewählt, welche durch das Finanz-Modell genutzt werden, um eine Abrechnung vorzunehmen. Sie fließen auch in die SLAs ein. Die Investitionskosten und Betriebskosten der Infrastruktur wurden in Anlehnung an Matros et al. (2009) [30] gewählt. Die Faktoren zur Bestimmung von Ausführungspreisen und Vertragsstrafen sind dem zur Zeit aktuellen Preisniveau des Weltmarktführers Amazon [4] nachempfunden. In Pasternak (2015) [33] wurden 9 verschiedene VM-Allocation-Policies vorgestellt und getestet. Die *MinimizeMigrationBestFit*-Strategie schnitt dabei hinsichtlich der Anzahl der Migrationen und verursachter Strafen am Besten ab und wird daher als Standard für alle Rechenzentren aller CSPs für weitere Simulationen gesetzt. Um die Größe der Rechenzentren zu bestimmen, werden Simulationen durchgeführt, in denen die Provider ohne an die Föderation angeschlossen zu sein ihre Workloads verarbeiten sollen. Aus diesen Simulationen ergeben sich eine untere und eine obere Schranke für die Anzahl der Hosts. Die untere Schranke markiert den Punkt, an dem der Provider seinen Workload nur noch so schlecht ausführen kann, dass seine Strafen für SLA-Brüche exponentiell ansteigen und höher sind als seine Einnahmen. Eine obere Schranke ergibt sich daraus, dass die Kosten der Infrastruktur ab einer Konfiguration größer als die Einnahmen sind. Um die Scheduling-Verfahren vergleichen zu können, muss eine Konfiguration gefunden werden, bei welcher genügend Migrationen auf Föderations-Ebene entstehen um einen Effekt beobachten zu können. Die Rechenzentren dürfen nicht zu groß konfiguriert sein. Andererseits können auch nicht alle Provider über nahezu ausgelastete Zentren verfügen. Dies würde zu einer Vielzahl an SLA-Strafen führen und die meisten Migrationsanfragen würden abgelehnt. Zwei Rechenzentren werden so dimensioniert, dass sie eine mittlere Größe aufweisen und

[3] Zur besseren Lesbarkeit wird auf die Nennung aller Parameter verzichtet

[4] zu finden unter `https://aws.amazon.com/de/ec2/pricing/`

die doppelte Menge der initial angefragten Ressourcen vorhalten, die übrigen Provider erhalten eine Menge an Hosts für ihre Rechenzentren, die im obersten Viertel der Spanne zwischen unterer und oberer Schranke liegen. Es ist anzunehmen, dass bei der Variation der Strategie auf DatacenterBroker-Ebene deutlich mehr Migrationen entstehen und die Föderation vor die Herausforderung stellen damit umzugehen. Insgesamt ergeben sich für die Föderation 176 Hosts mit 15244GHz CPU-Kapazität und 9240GB Arbeitsspeicher. Es werden 3297 VMs erzeugt, die verarbeitet werden müssen.

5.3 Simulation

5.3.1 Simulationsphase 1: Strategien auf Ebene der Rechenzentren

Im der ersten Simulationsphase sollen die Strategien auf der CSP-Ebene miteinander verglichen werden. Dazu erhalten alle Provider die oben beschriebene Grundkonfiguration und zunächst alle eine `FirstFit`-Strategie für die zweite Hierarchiestufe und eine `FairFit`-Strategie auf Föderationsebene. Diese Verfahren sind eher simpel gehalten und können daher gut zum Vergleichen genutzt werden. Bei einem Provider werden nun sukzessive alle `FederatedDatacenterBroker`-Strategien eingesetzt, um diese miteinander vergleichen zu können.

5.3.2 Auswertung Simulationsphase 1

Die simple `FirstFit`-Methode produziert für den gewählten Provider nur 3 Migrationen auf Föderations-Ebene. Ihre VM-Auswahl wird von der VM-Allocation-Policy getroffen, welche große VMs auswählt um die Migrationen gering zu halten. Nur zwei Strategien sind in der Lage einen höheren Profit zu erwirtschaften (siehe Abbildung 5.1) Durch die Veränderte VM-Auswahl können sie im eigenen Rechenzentrum mehr Einnahmen erwirtschaften und haben geringere Outsourcing-Kosten. Am profitabelsten erweist sich die MinimizeMigrationLocal-Strategie. Sie kann den Profit gegenüber der FirstFit um 1% steigern und ist 33% besser als die schlechteste hier gewählte Strategie und wird daher für die weitere Simulation selektiert.

5.3.3 Simulationsphase 2: Strategien auf Ebene der Föderation für einen CSP

Es werden im Folgenden die Strategien auf CSP-Ebene aller Föderationspartner fixiert und alle bis auf einen erhalten eine `FairFit`-CloudBroker-Strategie. Für diesen einen wird die Strategie gewechselt, so dass alle fünf Verfahren verglichen werden können.

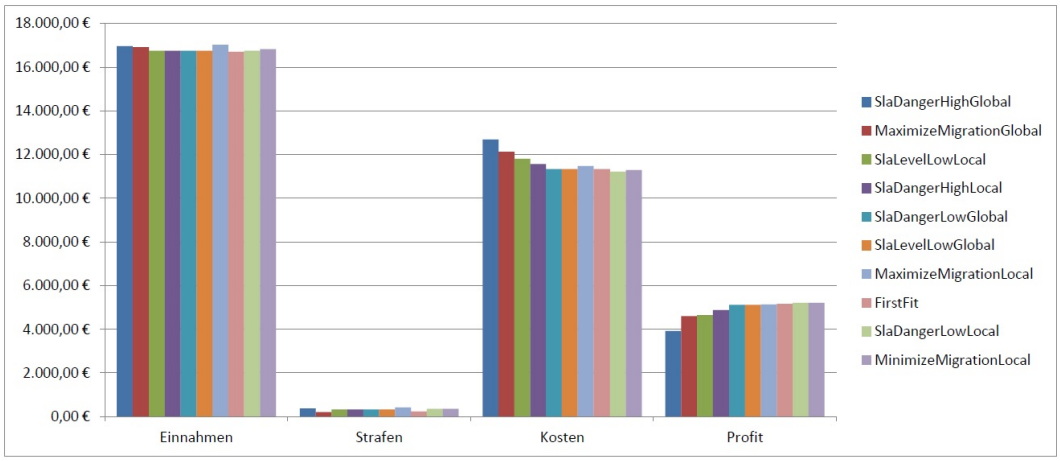

Abbildung 5.1: Vergleich der DC-Broker-Strategien

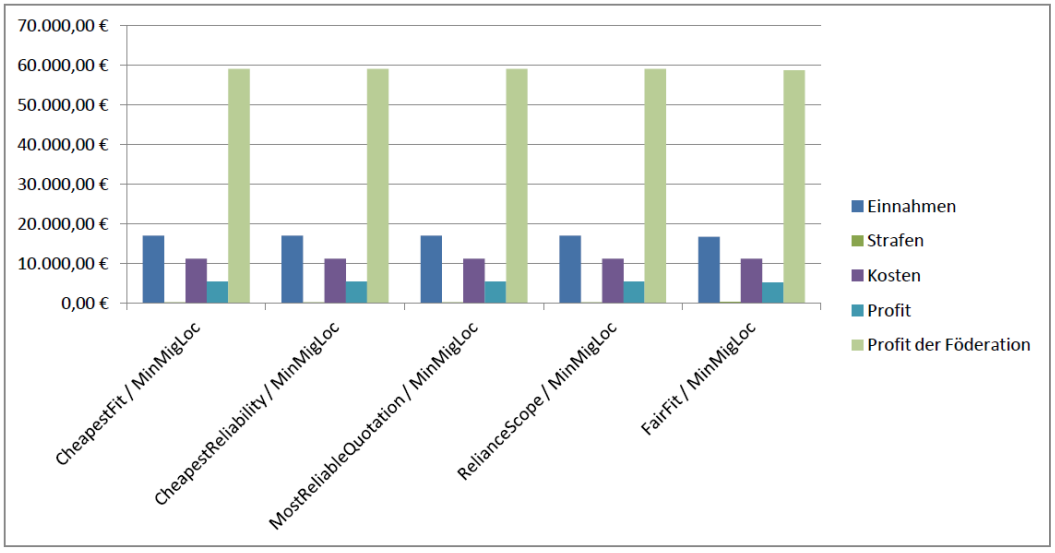

Abbildung 5.2: Vergleich der CB-Strategien

5.3.4 Auswertung Simulationsphase 2

Unabhängig von der gewählten Strategie wurden nahezu identische Ergebnisse erzielt (siehe Abbildung 5.2).

5.3.5 Simulationsphase 3: Strategien auf Ebene der Föderation

Alle Föderationspartner werden mit derselben Strategie auf Föderations-Ebene ausgestattet zunächst in der statischen und im Anschluss mit der dynamischeren Preisfindung durch Auktionen.

Gleiche Strategiekombination für alle CSP	Einnahmen der Föderation	Strafen der Föderation	Kosten der Föderation	Profit der Föderation
CheapestFit / MinMigLoc	154.494,21 €	3.855,00 €	91.797,87 €	58.841,33 €
RelianceScope / MinMigLoc	154.524,84 €	3.855,00 €	91.828,51 €	58.841,32 €
CheapestReliability / MinMigLoc	154.522,18 €	3.870,00 €	91.825,85 €	58.826,32 €
MostReliableQuotation / MinMigLoc	154.522,18 €	3.870,00 €	91.825,85 €	58.826,32 €

Gleiche Strategiekombination für alle CSP - Auktionsbasiert	Einnahmen der Föderation	Strafen der Föderation	Kosten der Föderation	Profit der Föderation
CheapestFit / MinMigLoc	154.328,28 €	3.855,00 €	91.630,15 €	58.843,12 €
CheapestReliability / MinMigLoc	154.328,28 €	3.855,00 €	91.630,15 €	58.843,12 €
MostReliableQuotation / MinMigLoc	154.328,28 €	3.855,00 €	91.630,15 €	58.843,12 €

Abbildung 5.3: Vergleich der CB-Strategien in der Föderation

5.3.6 Auswertung Simulationsphase 3

Auch in dieser Kombination wurden unabhängig von der gewählten Strategie nahezu identische Ergebnisse erzielt (siehe Abbildung 5.3).

5.4 Zusammenfassung der Ergebnisse

Trotz intensiver und vielfältiger Parameter- und Algorithmenauswahl konnte unter den gewählten Simulationskonfigurationen keine nennenswerten Verbesserungen erzielt werden. Es bleibt zu untersuchen, unter welchen Bedingungen die strategischen Vorteile der vorgestellten Algorithmen zum Tragen kommen.

Kapitel 6

Ausblick

Die Profitabilität von CSPs zu steigern, indem über ein sinnvolles Finanzmodell Ansatzpunkte zur Optimierung identifiziert werden, bleibt, trotz der Simulationsergebnisse dieser Arbeit ein interessanter Ansatz. In dieser Arbeit ist die Möglichkeit unberücksichtigt geblieben die Kosten der Infrastruktur zu senken. Dazu sind energieeffiziente Konsolidierungen notwendig. Es wurden dazu Vorbereitungen getroffen, um das Simulations-Framework um die Funktionalität zu erweitern, neben dem reaktiven Scheduling auch proaktiv ein Re-Scheduling durchzuführen. Dabei bietet sich auch die Möglichkeit Hosts zu konsolidieren. Es bleibt zu untersuchen, in wieweit der hier gewählte Ansatz nur in speziellen Lastsituation zur Anwendung kommen kann. Das FederatedCloudSim-Framework bietet dazu ideale Voraussetzungen. Insbesondere könnte eine genaure Protokollierung des Schedulings Aufschlüsse darüber geben, unter welchen Bedingungen eine Strategie erfolgversprechend ist. Insbesondere in dezentral organisierten Märkten liegt der Fokus darauf aus den sich bietenden lokalen Informationen sinnvolle Entscheidungen zu treffen. Abgelehnte Migrationsanfragen, zurückgeforderte und zurück migrierte VMs könnten solche Informationen liefern. Im kommerziellen Umfeld wird eine dezentrale Herangehensweise traditionell bevorzugt, um keine Unternehmens-Interna preiszugeben und die Handlungsvollmacht über die eigenen Ressourcen zu behalten. Zentrale Ansätze bieten allerdings den Vorteil eines effizienteren Schedulings. Es stellt folglich einen spannenden Forschungsansatz dar, ob es nicht Kompromisslösungen gibt, durch welche sich die Vorteile kombinieren lassen. Auch für eventuelle Forschung in diese Richtung, lässt sich das FederatedCloudSim-Framework anpassen, da die Modellierung von virtuellen CSPs möglich ist, die auch eine zentrale Vermittlerrolle einnehmen könnten.

Abbildungsverzeichnis

Listings

Literaturverzeichnis

[1] *Managing the Real Cost of On-Demand Enterprise Cloud Services with Chargeback Models*, 2010. https://www.techdata.com/content/tdcloud/files/cisco/Cloud_Services_Chargeback_Models_White_Paper.pdf.

[2] *Bitbrains Workload*, 2014. http://gwa.ewi.tudelft.nl/datasets/gwa-t-12-bitbrains.

[3] Baun, Christian, Marcel Kunze, Jens Nimis und Stefan Tai: *Cloud Computing*. Springer Berlin Heidelberg, 2010.

[4] Bedner, M.: *Cloud Computing: Technik, Sicherheit und rechtliche Gestaltung*. Forum Wirtschaftsrecht. Kassel University Press, 2013. https://books.google.de/books?id=3pyUqJog1eIC.

[5] Beloglazov, Anton und Rajkumar Buyya: *Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers*. Concurrency and Computation: Practice and Experience, 24(13):1397–1420, 2012.

[6] BITKOM: *Markt für Cloud Computing wächst ungebrochen*, 2014. https://www.bitkom.org/Presse/Presseinformation/Markt-fuer-Cloud-Computing-waechst-ungebrochen.html.

[7] BITKOM: *Cloud-Monitor 2016*, Mai 2015. https://www.bitkom.org/Presse/Anhaenge-an-PIs/2016/Mai/Bitkom-KPMG-Charts-PK-Cloud-Monitor-12-05-2016.pdf.

[8] Buyya, Rajkumar, Rajiv Ranjan und Rodrigo N Calheiros: *Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services*. In: *International Conference on Algorithms and Architectures for Parallel Processing*, Seiten 13–31. Springer, 2010.

[9] Buyya, Rajkumar, Chee Shin Yeo, Srikumar Venugopal, James Broberg und Ivona Brandic: *Cloud computing and emerging {IT} platforms: Vision, hype,*

and reality for delivering computing as the 5th utility. Future Generation Computer Systems, 25(6):599 – 616, 2009.

[10] Cafaro, Massimo und Giovanni Aloisio: *Grids, Clouds, and Virtualization.* In: *Grids, Clouds and Virtualization*, Seiten 1–21. Springer, 2011.

[11] Calheiros, Rodrigo N., Rajiv Ranjan, Anton Beloglazov, César A. F. De Rose und Rajkumar Buyya: *CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms.* Softw. Pract. Exper., 41(1):23–50, Januar 2011.

[12] Clearwater, Scott H: *Market-based control: A paradigm for distributed resource allocation.* World Scientific, 1996.

[13] Copeland, Thomas E, Tim Koller und Jack Murrin: *Unternehmenswert: Methoden und Strategien für eine wertorientierte Unternehmensführung.* Campus Verlag, 2002.

[14] Dong, Fangpeng und Selim G Akl: *Scheduling algorithms for grid computing: State of the art and open problems.* Technischer Bericht, Technical report, 2006.

[15] Foster, Ian, Yong Zhao, Ioan Raicu und Shiyong Lu: *Cloud computing and grid computing 360-degree compared.* In: *2008 Grid Computing Environments Workshop*, Seiten 1–10. Ieee, 2008.

[16] Geelan, Jeremy: *Twenty-One Experts Define Cloud Computing*, 2009. `http://cloudcomputing.sys-con.com/node/612375`.

[17] Goiri, Inigo, Jordi Guitart und Jordi Torres: *Characterizing cloud federation for enhancing providers' profit.* In: *2010 IEEE 3rd International Conference on Cloud Computing*, Seiten 123–130. IEEE, 2010.

[18] Goll, Joachim: *Architektur- und Entwurfsmuster der Softwaretechnik*, Band 2. Springer Vieweg, 2014.

[19] Gondek, Verena: *Hybrid Flow-Shop Scheduling mit verschiedenen Restriktionen: Heuristische Lösung und LP-basierte untere Schranken.* Doktorarbeit, Universität Duisburg-Essen, Fakultät für Mathematik, 2011.

[20] Graham, Ronald L, Eugene L Lawler, Jan Karel Lenstra und AHG Rinnooy Kan: *Optimization and approximation in deterministic sequencing and scheduling: a survey.* Annals of discrete mathematics, 5:287–326, 1979.

[21] Huberman, Bernardo A et al.: *The ecology of computation.* North-Holland Amsterdam, 1988.

[22] Kafil, M. und I. Ahmad: *Optimal task assignment in heterogeneous distributed computing systems.* IEEE Concurrency, 6(3):42–50, Jul 1998.

[23] Kalra, Mala und Sarbjeet Singh: *A review of metaheuristic scheduling techniques in cloud computing.* Egyptian Informatics Journal, 16(3):275–295, 2015.

[24] Kecskemeti, Gabor, Attila Kertesz, Attila Marosi und Peter Kacsuk: *Interoperable resource management for establishing federated clouds.* Achieving Federated and Self-Manageable Cloud Infrastructures: Theory and Practice, 2:18–35, 2012.

[25] Kecskeméti, Gábor, Attila Kertész, Attila Marosi und Zsolt Németh: *Strategies for Increased Energy Awareness in Cloud Federations.* 2013.

[26] Kohne, Andreas: *Modell für ein SLA-basiertes VM-Scheduling in föderierten Cloud-Umgebungen.* In: *GI-Jahrestagung*, Seiten 3047–3061, 2013.

[27] Kohne, Andreas und Olaf Spinczyk: *Model for SLA-Based VM Scheduling in Federated Cloud Environments.* Journal of Integrated Design and Process Science, 18(1):39–52, 2014.

[28] Kohne, Andreas, Marc Spohr, Lars Nagel und Olaf Spinczyk: *FederatedCloudSim: A SLA-aware Federated Cloud Simulation Framework.* In: *Proceedings of the 2Nd International Workshop on CrossCloud Systems*, CCB '14, Seiten 3:1–3:5, New York, NY, USA, 2014. ACM.

[29] Kunesch, U, B Reti und M Pauly: *White Paper Cloud Computing.* 2009.

[30] Matros, Raimund, Philipp Stute und Nicolaus Heereman von Zuydtwyck: *Make-or-Buy im Cloud-Computing-Ein entscheidungsorientiertes Modell für den Bezug von Amazon Web Services.* 2009.

[31] May, Michael et al.: *Sammlung und Nutzung freier Ressourcen in Weitverkehrsnetzen.* Doktorarbeit, Technische Universität München, 2000.

[32] Mell, Peter und Tim Grance: *The NIST definition of cloud computing.* 2011.

[33] Pasternak, Damian: *Implementierung und Evaluation von Scheduling-Strategien in FederatedCloudSim.* Diplomarbeit, TU Dortmund, 2015.

[34] Patel, Pankesh, Ajith H Ranabahu und Amit P Sheth: *Service level agreement in cloud computing.* 2009.

[35] Pinedo, Michael: *Scheduling.* Springer, 2015.

[36] Schneider, Matti und Sebastian Mentemeier: *Zeitreihenanalyse mit R*, 2016.

[37] Sekhar, Jyothi, Getzi Jeba und S Durga: *A survey on energy efficient server consolidation through vm live migration.* International Journal of Advances in Engineering & Technology, 5(1):515–525, 2012.

[38] Spohr, Marc: *FederatedCloudSim: Ein Simulationsframework für SLA-basierte, föderierte Cloud-Umgebungen.* Diplomarbeit, TU Dortmund, 2015.

[39] Tanenbaum, Andrew S und Maarten van Steen: *Verteilte Systeme: Grundlagen und Paradigmen*, Band 1. Pearson Studium, 2003.

[40] Teng, Fei: *Ressource allocation and schelduling models for cloud computing.* Theses, Ecole Centrale Paris, Oktober 2011.

[41] Wellman, Michael P, William E Walsh, Peter R Wurman und Jeffrey K MacKie-Mason: *Auction protocols for decentralized scheduling.* Games and economic behavior, 35(1):271–303, 2001.

[42] Wieder, Philipp, Joe M. Butler, Wolfgang Theilmann und Ramin Yahyapour: *Service Level Agreements for Cloud Computing.* Springer Science+Business Media, LLC, 2011.

[43] Wikimedia: *XaaS.* `https://commons.wikimedia.org/w/index.php?curid=36505422`.

[44] Wolfstetter, Elmar: *Auktionen und Ausschreibungen - Bedeutung und Grenzen des „linkage“–Prinzips.* Nummer 31 in *Sonderforschungsbereich 373: Quantification and Simulation of Economic Processes.* Wirtschaftswissenschaftliche Fakultät, 1998.

[45] Zahn, Annja: *Elektronisches Handeln mit Rechenleistung.* Wissner, 1999.

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet sowie Zitate kenntlich gemacht habe.

Dortmund, den 4. Oktober 2016

Marcel Krüger