Modell für ein SLA-basiertes VM-Scheduling in föderierten Cloud-Umgebungen

Dipl. Inf. Andreas Kohne
(andreas.kohne@gmx.de)
Technische Universität Dortmund, MATERNA GmbH

Abstract: Zukünftig werden sich immer mehr *Cloud Service Provider* zu *Cloud-Födera-tionen* zusammenschließen, um zum einen gemeinsam mehr *Services* für die Kunden anzubieten und zum anderen ihre Ressourcen bestmöglich auszulasten. Grundlage dieser Föderationen sind *Service Level Agreements* (*SLAs*), die die Güte der zu erbringenden Dienste zwischen den Anbietern untereinander sowie zwischen den Anbietern und den Kunden regeln. Eine große Herausforderung ist das verteilte, *SLA-basierte Scheduling* von virtuellen Maschinen (*VMs*). Hierzu wird ein allgemeines Modell zur Untersuchung von verschiedenen *VM-Scheduling-Strategien* in verteilten *Cloud-Föderationen* entwickelt. Das Modell kann als Schablone eingesetzt werden, um mit Hilfe des vorgestellten *Multi-Level-Schedulers* komplexe *Cloud-Föderations-Szenarien* zu untersuchen.

1 Einführung

Cloud Computing gewinnt zunehmend an Bedeutung und die Marktakzeptanz steigt ungebremst (vgl. [idc11]). Vor allem die *Cloud-basierte* Bereitstellung von *Software*, als *Software as a Service* (*SaaS*) bezeichnet, hat sich sehr gut im Markt etabliert. Doch auch die weiteren *Services* wie *Platform as a Service* (*PaaS*) oder *Infrastructure as a Service* (*IaaS*) gewinnen an Bedeutung. Zurzeit setzen viele Unternehmen noch auf die Bereitstellung von *Cloud-Services* aus dem eigenen Rechenzentrum (*RZ*). Dies wird als *Private Cloud* bezeichnet. Den lokalen *Services* werden aber immer mehr auch *Services* von globalen *Cloud Service Providern* (*CSPs*) aus der *Public Cloud* beigemischt. Hierbei wird von einer *Hybrid Cloud* gesprochen.

Als Basis für die flexible Bereitstellung von *Cloud-Diensten* hat sich neben der Standardisierung und Automatisierung der zugrunde liegenden IT-Infrastruktur die Virtualisierung als Basistechnologie in den *Cloud-Rechenzentren* durchgesetzt. Beginnend bei der Speicherbereitstellung, über die Netzwerke bis hin zu den virtuellen Maschinen (*VMs*) werden alle IT-Ressourcen mit Hilfe von Technologie-spezifischen Methoden virtualisiert, konsolidiert und bedarfsgerecht zur Verfügung gestellt.

Eine effiziente Bereitstellung von *VMs* auf einem Verbund mehrerer *Server*, einem sogenannten *Cluster*, in einem *RZ* erfordert spezielle *Scheduling-Algorithmen* auf *Hypervisorebene*, die auf die jeweiligen Bedürfnisse des Betreibers besonders angepasst werden. Hier können Hochverfügbarkeit, Lastausgleich, Kosten- oder Energieeffizienz im Mittelpunkt

stehen. Solche Algorithmen sind bereits für klassische *RZs* gut erforscht. Weltweit verteilte *RZs*, die zu einem *CSP* gehören und potentiell die gleichen *Services* anbieten können, stellen die Forschung aktuell aber wieder vor Herausforderungen.

Neben den großen *Cloud-Anbietern* spielen immer mehr auch kleine *CSPs* eine Rolle. Sie spezialisieren sich oft auf Branchen-spezifische Angebote oder Nischenlösungen. Eine Möglichkeit für kleine *CSPs*, trotzdem ein breites *Service-Angebot* für ihre Kunden bereitzustellen, ist es, sich mit anderen *CSPs* zusammenzuschließen und die Dienste des Anderen in Gänze oder in Teilen über entsprechende Kooperationsverträge mit anzubieten. Hierbei wird von einer *Cloud-Föderation* gesprochen.

Basis einer solchen *Cloud-Föderation* sind sogenannte *Service Level Agreements* (*SLAs*). Sie beschreiben in einem Vertrag die jeweils bereitgestellten Ressourcen und *Services*, deren *Service-Parameter* wie zum Beispiel Erreichbarkeit, *Service-Zeiten* oder Datendurchsatz und die bei der Nutzung entstehenden Kosten. Weiterhin werden Verantwortlichkeiten bei Nichterbringung von vereinbarten *Services* und eventuelle Strafen, sogenannte Pönalen, geregelt.

Ein verteiltes, *SLA-basiertes Cloud-Scheduling* ist bisher wenig erforscht. Deshalb wird in diesem Beitrag ein allgemeines Modell zur Untersuchung von *SLA-basiertem VM-Scheduling* in föderierten *Cloud-Umgebungen* vorgestellt. Das Modell ist so konzipiert, dass es als Schablone für viele verschiedene Versuche eingesetzt werden kann. Dazu wird ein verteiltes *Multi-Level-Scheduling-Verfahren* beschrieben, welches auf den verschiedenen Entscheidungsebenen Schnittstellen anbietet, über die verschiedenste *Scheduling-Strategien* eingebunden werden können.

Das restliche Dokument gliedert sich wie folgt: Abschnitt 2 zeigt den aktuellen Stand zum Thema *VM-Scheduling* in *Cloud-Umgebungen* auf. Danach wird in 3 der Einsatz von *SLAs* in *Cloud-Umgebungen* erläutert. Weiterhin werden in 4 der Aufbau und die Einsatzmöglichkeiten von *Cloud-Föderationen* beschrieben. Im Anschluss daran wird in Abschnitt 5 beschrieben, wie anhand eines realitätsnahen Modells von *Cloud-Föderationen SLA-basiertes VM-Scheduling* untersucht werden kann. Danach werden die Ergebnisse in Abschnitt 6 diskutiert. Abschnitt 7 beinhaltet eine Zusammenfassung und einen Ausblick.

2 VM-Scheduling in Cloud-Umgebungen

Um die verfügbaren *Server-Ressourcen*, wie z.B. Hauptspeicher oder Prozessorzeit, auf die verschiedenen *VMs* aufzuteilen, besitzt jeder *Hypervisor* einen individuell anpassbaren *Scheduler*. In *Cloud-RZs* stehen viele *Server-Systeme* in einem oder mehreren *Clustern* zusammengefasst zur Verfügung, auf denen potentiell die *VMs* ausgeführt werden können. Es gibt also einen zentralen *Scheduler* in jedem *RZ*, der zwei Hauptaufgaben besitzt:

1. *Initiale VM-Platzierung:* Wenn eine neue *VM* gestartet werden soll, muss der *Scheduler* anhand seiner jeweiligen Strategie entscheiden, auf welchem physikalischen *Server* die *VM* ausgeführt werden soll.
2. *VM-Migration:* Im laufenden Betrieb muss der *Scheduler* anhand seiner Strate-

gie entscheiden, ob *VMs* mit Hilfe von *Live-Migrationen* zwischen verschiedenen physikalischen *Servern* verschoben werden sollen. Dies kann z.B. bei einem Lastausgleich der Fall sein.

VM-Schedulings sind in vielen Bereichen bereits erforscht. Einen guten Überblick über aktuelle *Cloud-Scheduling-Algorithmen* geben Teng [Ten11] und Tilak et. al. [TP12]. Dort werden verschiedene Herangehensweisen von Markt-basierten über heuristische und Echtzeit-Algorithmen bis hin zu Partikelschwarm- und Spieltheoretischen-Ansätzen beleuchtet.

VM-Scheduler entscheiden anhand von verschiedensten Strategien. Wird nur ein *RZ* betrachtet, so kann ein mögliches Ziel des *Schedulers* z.B. ein gleichmäßiger Lastausgleich aller physikalischen *Server* sein. Weiterhin kann der *Scheduler* auch versuchen, besonders Energie-effizient zu sein, in dem er die *VMs* auf möglichst wenigen *Servern* mit Hilfe von *Live-Migrationen* zusammenfasst. Danach können nicht benötigte *Server* in einen Energiesparmodus versetzt oder ganz herunterzufahren werden (vgl. [VT10]). Einen guten Überblick zum Energie-effizienten *VM-Scheduling* liefern Sekhar et. al. in [SJD12]. Weitere *Scheduling-Ziele* können u.A. Sicherheit, Kosteneffizienz, Optimierung des Daten- oder Netzwerkdurchsatzes oder *SLA-Prioritäten* sein.

In *Cloud-Szenarien*, in denen *CSPs* nicht nur ein *RZ* besitzen, sondern potentiell mehrere über die Welt verteilte Lokationen, müssen die *Scheduler* diesen Aspekt natürlich mit einbeziehen. Hier kommen aber weitere Herausforderungen, wie z.B. der Datendurchsatz und die Verzögerung der *VM-Migration*, mit hinzu. Weiterhin ist es teilweise nicht möglich, laufende *VMs* über sehr lange Distanzen zu migrieren. Hier sind also auch technologische Limitierungen mit einzubeziehen.

In verteilten *Cloud-Umgebungen*, in denen *VMs* sogar bei einem anderen *CSP* betrieben werden können, werden noch komplexere Strategien nötig. Hier wird grundsätzlich zwischen zwei verschiedenen *Scheduling-Varianten* unterschieden:

1. *Meta-Scheduler:* Der *Meta-Scheduler* hat alle Informationen zu den angeschlossenen *CSPs* zur Verfügung. Hierzu gehören z.B. Daten über die aktuelle Lastsituation oder die noch zur Verfügung stehenden Ressourcen der jeweils angeschlossenen *CSPs*. *Meta-Scheduler* sind bereits im Umfeld des *Grid-Computing* gut erforscht und die Ergebnisse sind in vielen Bereichen auf das *Cloud Computing* übertragbar. Beispielsweise schlagen Brandic et. al. in [BVMB08] einen *Grid-Meta-Scheduler* vor, der vier zentrale Aufgaben übernimmt: 1. *Publish*: Bereitstellen von Informationen, 2. *Lookup*: Suche nach potentiellen Ressourcen, 3. *Match*: Finden des Anbieters, der die Anforderungen bestmöglich erfüllt, und 4. *Negotiate*: Aushandeln von Leistungen und Kosten. Der Ansatz eines *Meta-Schedulers* ist aber für kommerzielle *Clouds* nicht verbreitet, da jeder *CSP* seine eigenen *Scheduling-Strategien* besitzt und aus wettbewerbstechnischen Gründen oft keine Lastdaten öffentlich macht.

2. *Verteilter Ansatz:* Bei dem verteilten *Scheduling-Ansatz* besitzt jeder *CSP* einen lokalen *Scheduler*, der für das *Scheduling* der *VMs* in seinem *RZ* zuständig ist. Der *Scheduler* kann aber auch entscheiden, dass eine *VM* bei einem anderen *CSP* ausgeführt werden soll. Dazu kann er über eine definierte Schnittstelle Anfragen direkt

an einen oder mehrere *CSPs* stellen. Er bewertet die Ergebnisse dann lokal und trifft eine entsprechende Entscheidung. Diese Variante ist für den Bereich der föderierten *Clouds* sehr gut geeignet. Das im weiteren Verlauf dieses Beitrags beschrieben Modell basiert auf dieser verteilten *Scheduling-Variante*.

3 Service Level Agreements

SLAs werden oft im Bereich der IT-basierten *Services* eingesetzt, um die Dienstgüte einer *Service-Leistung* zu beschreiben und vertraglich festzulegen. In einem *SLA* werden die funktionalen und nicht-funktionalen Anforderungen an einen speziellen *Service* beschrieben. Eine funktionale Anforderung könnte zum Beispiel die Ausführung einer *VM* mit einem speziellen Betriebssystem und einem *Webserver* in einer bestimmten Konfiguration sein. Die nicht-funktionalen Anforderungen könnten z.B. die durchschnittliche Erreichbarkeit des *Webservers* über das *Internet* oder eine zugesicherte Rechenzeit auf einer *CPU* sein. Um diese Parameter überprüfbar zu machen, werden für jede Anforderung sogenannte *Quality of Service* (*QoS*) *Level* definiert, welche einzeln messbar und damit nachweisbar sind. Ein Kunde kann also einen oder mehrere *Services* bei einem Dienstleister bestellen, über die er jeweils ein *SLA* abschließt. Der Kunde kann dann den *Service* nutzen und der Anbieter muss zu jeder Zeit nachweisbar belegen, dass der entsprechende *Service* in der vereinbarten Dienstgüte zur Verfügung gestanden hat (vgl. [Ber07]).

Der Einsatz von *SLAs* für die Bereitstellung von *IT-Services* ist in vielen Bereichen bereits erforscht. So beschrieben Ching et. al. in [CSM03] z.B. den Einsatz von *SLAs* im *Grid Computing*. Sie beschrieben ein Vorgehen, in dem sie *IT-Ressourcen* modellieren und über eine Beschreibungssprache *Service-Garantien* für die jeweilige Nutzung geben können. Die so erstellten *SLAs* können dann automatisch interpretiert und in Ressourcenreservierungen umgewandelt werden. Auch hier können viele Forschungsergebnisse aus dem Bereich des *Grid Computings* für das *Cloud Computing* adaptiert werden.

Wichtig für den effektiven Einsatz von *SLAs* im *Cloud Computing* ist, dass sich die *SLAs* in maschinenlesbarer Form repräsentieren lassen. Nur dadurch können sie automatisch verarbeitet und überprüfbar gemacht werden. In den letzten Jahren haben sich dazu mehrere Beschreibungssprachen herausgebildet. Im Folgenden werden die beiden bekanntesten mit je einer konkreten Umsetzung vorgestellt.

WSLA: Patel et. al. nutzen die *XML-basierte* Beschreibungssprache *WSLA* (*Web Service Level Agreement*), um *SLAs* im *Cloud Computing* zu repräsentieren und durchzusetzen. Sie gehen in [PRS09] aber davon aus, dass das *SLA* bereits abgeschlossen ist. Somit wird der Bereich des Aushandelns ausgeblendet. Sie beschreiben weiterhin eine Möglichkeit, *SLAs* aufzusplitten, damit in dem Fall, dass ein *Service* von mehreren Dienstleistern gemeinsam erbracht wird, nicht jede Partei alle Informationen aus dem Vertrag erhält.

WS Agreement: In dem Forschungsprojekt *SLA@SOI* wurde die zweite der beiden meistgenutzten Beschreibungssprachen für *SLAs*, *WS Agreement*, eingesetzt, um die Dienst-

güte zu beschrieben. Comuzzi et. al. haben darauf aufbauend ein eigenes *SLA-Framework* entwickelt, welches sich aus einem eigenen *SLA-Modell*, einer kompletten *Management Referenzarchitektur* und einer *Monitoring-Integration* zusammensetzt (vgl. [CKR+10]). Der Fokus der Arbeit liegt darauf, *SLAs* zu beschreiben, zu verhandeln, auszutauschen und durchsetzen. Hierbei wird aber nur auf die *SLAs* zwischen Kunden und Anbietern eingegangen. Der Aspekt einer *Cloud-Föderation*, in der auch *CSPs* untereinander *SLAs* abschließen können, wird nicht weiter betrachtet.

SLAs haben sich als zentrale Vertragsgrundlage von *Cloud-Services* durchgesetzt. Sie eigenen sich zum einen zur konkreten Beschreibung von Anforderungen an einen *Cloud-Service*. Zum anderen kann die Dienstgüte eines aktiven *Services* mithilfe eines *SLAs* automatisch überwacht werden. *SLAs* bilden weiterhin die Grundlage für *Cloud-Föderationen*. Darum werden im weiteren Verlauf des Beitrags *SLAs* als zentrales Verwaltungselement eingesetzt.

4 Cloud Föderationen

Das Paradigma des *Cloud Computings* hat sich in den letzten Jahren eindeutig durchgesetzt. Viele große Anbieter stellen bereits *Cloud-Dienste* in einer *Public Cloud* zur Verfügung. Lokale, oder auch *Private Clouds*, sind ebenfalls stark verbreitet. Die Vorbehalte in den Bereichen Sicherheit, Vertrauen und Erreichbarkeit zerstreuen sich immer mehr, nicht zuletzt durch die starke Nachfrage aus dem Sektor der Privatanwender. Zurzeit gibt es noch viele unterschiedliche *Cloud-Infrastrukturen*. Zum Beispiel hat sich noch kein *Hypervisor* klar als Standard durchgesetzt. Dies führt dazu, dass *VMs*, welche bei einem *CSP* betrieben werden, meist nicht kompatibel mit der *Cloud-Infrastruktur* eines Mitbewerbers sind. Dies ist ein klarer Nachteil für die Kunden, da sie sich meist auf einen Anbieter festlegen müssen. Der Grad der Standardisierung im Bereich des *Cloud Computings* muss daher weiter vorangetrieben werden.

In Zukunft wird es immer mehr *Cloud-Föderationen* geben. Eine *Cloud-Föderation* ist ein Zusammenschluss mehrerer *CSPs* zu dem Zweck, gemeinsam mehr *Services* für die Endkunden anzubieten und z.B. bei Lastspitzen *Services* zu Föderationspartnern auszulagern. Ein entsprechender Zusammenschluss erlaubt es den einzelnen *CSPs*, die angebotenen *Services* eines oder mehrerer kooperierender *CSPs* in Gänze oder in Teilen mit anzubieten. Dies ermöglicht es den *CSPs*, ihr Leistungsportfolio um weitere Dienstleistungen zu erweitern, ohne dass sie diese selbst erbringen müssen. Vor allem kleinere *CSPs*, die sich auf die Bereitstellung von Branchenlösungen oder Nischenprodukten spezialisiert haben, profitieren von solch einem Zusammenschluss. Grundlage einer *Cloud-Föderation* ist ein über *SLAs* geregelter Vertrag, der beschreibt, welcher *CSP* welche *Services* in welcher Dienstgüte zu welchen Preisen zur Verfügung stellt.

Wie bereits in 2 beschrieben, sind zwei Verwaltungsarten für Cloud-Föderationen möglich, die in den nächsten beiden Abschnitten beschrieben werden. Hierbei wird ab jetzt vereinfachend angenommen, dass eine *VM* gleichbedeutend mit einem angebotenen *Service* ist

(z.B. *Webserver*, *Datenbankserver* usw.) und als Ganzes innerhalb eines *Cloud-RZs* oder zwischen verschiedenen *CSPs* migriert werden kann. Natürlich kann ein *Service* auch aus mehreren *VMs* bestehen, die entweder immer zusammen in einem *RZ* ausgeführt werden müssen, oder auch über mehrere *RZs* oder sogar mehrere *CSPs* verteilt sein können.

4.1 Zentraler Ansatz

Hierbei gibt es eine zentrale Entscheidungsinstanz, die alle angeschlossen *CSPs* verwaltet und für die Zuweisung von *VMs* (*Services*) auf die verschiedenen *RZs* der *CSPs* zuständig ist. Diese Instanz kennt alle angeschlossenen *CSPs*, ihre in der Föderation angebotenen Dienste, die entsprechenden *SLAs* und die dafür veranschlagten Kosten. Kunden können *Service-Anfragen* (*Service Requests*) an diese zentrale Instanz schicken, in der sie einen oder mehrere *Services* in einer bestimmten Dienstgüte anfragen, und erhalten eine Liste mit *CSPs*, die in der Lage wären, den *Service* zu erbringen. Der Kunde kann dann manuell oder automatisch anhand seiner eigenen Strategie entscheiden, welcher *CSP* den *Service* erbringen soll. Natürlich ist es auch möglich, dass die zentrale Entscheidungsinstanz den bestmöglichen Anbieter selbstständig auswählt. Dazu muss der Kunde in seiner Anfrage die *Service-Kriterien* entweder sehr genau spezifizieren, oder Entscheidungsspielräume vorgeben.

Cloud-Föderationen sind aktuell Inhalt vieler Forschungsprojekte. Hierbei werden die unterschiedlichsten Aspekte, wie z.B. unterschiedliche *VM-Scheduling-Ansätze*, *SLA-Verhandlungs-* und *Management-Ansätze* und verschiedenste Optimierungen, z.B. von Datendurchsatz, oder Energieverbrauch, untersucht. Die meisten der im Folgenden beschriebenen Forschungsprojekte setzen dabei auf Simulationen. Das flexibel erweiterbare *Simulationsframework CloudSim* wird dabei oft als Basis eingesetzt. Calheiros et. al. beschreiben das *Framework* in [CRB+11] ausführlich. Nachfolgend werden einige Forschungsprojekte, die einen zentralen *Scheduling-Ansatz* nutzen, vorgestellt.

In ihrer Veröffentlichung [BRC10] schlagen Buyya et. al. einen föderierten *Cloud-Ansatz* vor, den sie *InterCloud* nennen. Jeder *CSP* benötigt hierbei eine Komponente namens *Cloud Coordinator*, um an der Föderation teilnehmen zu können. Nutzer, die einen *Service* der angeschlossenen *CSPs* nutzen wollen, müssen einen speziellen *Cloud Broker* nutzen, um Anfragen an einen zentralen Marktplatz, den so genannten *Cloud Exchange*, stellen zu können. Diese zentrale Komponente kennt wiederum alle *CSPs* und ihre *Services*. Kundenanfragen werden vom *Cloud Exchange* angenommen und nach definierten Kriterien dem *CSP* mit dem besten Angebot zugewiesen. Bei ihren Untersuchungen wurden bisher nur die Initialplatzierungen der *VMs* untersucht. Die Untersuchungen anhand des *CloudSim Frameworks* haben gezeigt, dass durch den Einsatz einer *Cloud-Föderation*, die Berechnungs- und Wartezeiten der *VMs* deutlich gegenüber der reinen Ausführung in einem *Cloud-RZ* reduziert werden konnten.

Jrad et. al. untersuchen in [JTS12b] einen zentralen *Cloud Broker*, welcher *SLA-basiert VMs* unterschiedlichen *CSPs* zuweisen kann. Dabei kennt der *Broker* alle angeschlossenen *CSPs*, ihre angebotenen *Services* und die jeweiligen *SLAs*. Weiterhin wird in den Unter-

suchungen nur die Erstplatzierung der *VMs* in den jeweiligen *CSP-RZs* ausgewertet. Eine Überprüfung des gesamten *SLA-Lifecycles*, sprich eine Überwachung der *VMs* über ihre gesamte Laufzeit, ist bisher nicht vorgesehen. Um ihre Ideen zu überprüfen, setzen sie das bereits erwähnte *CloudSim* mit entsprechenden Anpassungen ein (vgl. [JTS12a]). Auch sie können zeigen, dass der Einsatz einer *Cloud-Föderation* sich positiv auf die Verteilung der *VMs* auswirkt.

Celesti et. al. schlagen in [CTVP10] einen *Cross-Cloud Federation Manager* vor. Er stellt das zentrale Bindeglied der Föderation dar und übernimmt drei Aufgaben. 1. *Discover:* Hierbei werden mit einem *Peer-to-Peer-Ansatz* mögliche Föderationspartner gesucht. 2. *Match Making:* Dieser Prozess sucht aus den gefundenen Kooperationspartnern den bestmöglichen anhand von vordefinierten Kriterien heraus. 3. *Authentification:* Das System sorgt dafür, dass der Kunde bei der entsprechenden *Cloud* per *Single-Sign-on* authentifiziert wird, um danach *VMs* bestellen zu können.

In diesem Vorschlag werden keine *SLAs* berücksichtigt und Entscheidungen nur anhand von einigen festen Kriterien vorgenommen, die statisch miteinander verglichen werden. Ebenfalls wird wieder nur die initiale Verteilung der *VMs* mit Hilfe einer zentralen Entscheidungsinstanz untersucht und nicht der ganze Lebenszyklus der Systeme betrachtet.

4.2 Verteilter Ansatz

In diesem Ansatz gibt es keine zentrale Verwaltungsinstanz. Jeder *CSP* verwaltet die *VMs* innerhalb seiner *RZs* selber. Trotzdem kann er über die Föderation *VMs* zu kooperierenden *CSPs* migrieren, oder sie direkt dort starten, falls er den entsprechenden *Service* gar nicht selbst anbietet. Dieser Ansatz gibt den *CSPs* maximale Flexibilität, da jeder *CSP* entscheiden kann, mit welchen *CSPs* er eine Kooperation unterhalten will. Ebenso können zwischen unterschiedlichen *CSPs* unterschiedliche *SLAs* ausgehandelt werden, was über einen zentralen Ansatz nur schwer abbildbar ist. Außerdem kann hierbei jeder *CSP* seine eigenen Entscheidungsstrategien zur Annahme, Weiterleitung oder Ablehnung eines *Service Requests* implementieren. Ein Nachteil dieser Lösung ist es, dass Kunden ihre *Service Requests* nicht mehr an einer zentralen Stelle einreichen können, sondern ein Vertragsverhältnis mit einem oder potentiell mehreren *CSPs* eingehen müssen.

In dem internationalen Forschungsprojekt *EASI-CLOUDS* (*Extensible Architecture and Service Infrastructure for Cloud-aware Software*) wird solch ein verteilter Ansatz für föderierte *Cloud-Umgebungen* untersucht (vgl. [eas13]). Das Projekt hat zum Ziel, eine *Open-Source-basierte Cloud-Infrastruktur-Software* zur Verfügung zu stellen. Dabei wird besonderes Augenmerk auf die Bereitstellung von standardisierten Schnittstellen für einen *Service-Austausch* gelegt. Weiterhin wird ein *Orchestrierungsframework* entwickelt, welches verteilte *Services* automatisch erstellen und überwachen kann. Integraler Bestandteil der Lösung ist eine Komponente zur Kapazitätsplanung, Verwaltung und Abrechnung von *Cloud-Services*. Basis der Föderation und zentrales Mittel zur Erstellung und Verwaltung der *Services* sind *SLAs*. Dazu wird in dem Projekt ein *SLA-basierter* Verwaltungsansatz erforscht, der den ganzen Lebenszyklus eines *Services* abbildet.

Der im weiteren Verlauf dieses Beitrags vorgestellte *SLA-basierte VM-Scheduling-Ansatz* für föderierte *Cloud-Umgebungen* ist Teil der Forschung aus dem *EASI-CLOUDS* Projekt.

5 Modell für ein SLA-basiertes VM-Scheduling in föderierten Cloud-Umgebungen

Nachdem in den letzten Abschnitten die Grundlagen für *SLA-basierte Cloud-Föderationen* beschrieben wurden, wird in diesem Abschnitt ein abstraktes Föderationsmodell vorgestellt. Anhand dieses Modells kann verteiltes *VM-Scheduling* unter Berücksichtigung von *SLAs* untersucht werden. Dazu wird im Folgenden zuerst der Aufbau der Föderation erläutert. Darauf wird beschrieben, wie *SLAs* in das Modell integriert werden. Im Anschluss daran wird ein *Template* für einen verteilten *Multi-Level-VM-Scheduler* vorgestellt, der unter Berücksichtigung der geschlossenen *SLAs* die Verteilung der *VMs* in der Föderation übernimmt.

5.1 Komponenten des Föderationsmodells

Um ein möglichst realitätsnahes Modell einer *Cloud-Föderation* abzubilden, werden zwei grundlegende Föderationskomponenten benötigt, die im Folgenden erläutert werden:

1. *Cloud Service Provider:* Ein *CSP* kann grundsätzlich ein oder beliebig viele *RZs* besitzen. Jedes *RZ* verfügt über eine feste Anzahl von *Servern*, die die Ressourcen für die *VMs* zur Verfügung stellen. Unterschiedliche *RZs* können unterschiedlich viele, unterschiedlich konfigurierte *Server* besitzen. Jeder *CSP* stellt mindestens einen *Service* für Kunden in Form von einer oder mehreren *VMs* zur Verfügung. Bucht ein Kunde einen solchen *Service*, werden die entsprechenden *VMs* für ihn gestartet. Unterschiedliche *RZs* eines *CSPs* können unterschiedliche *Services* anbieten. *CSPs* können ihre *Services* direkt oder über einen *virtual Cloud Service Provider* (*vCSP*) den Kunden anbieten.
2. *virtual Cloud Service Provider: vCSPs* besitzen keine eigenen Ressourcen im Sinne von *Servern*, die *VMs* ausführen können. Sie sind als Dienstleister zu verstehen, die Verträge mit *CSPs* besitzen und deren Dienste bündeln und Kunden anbieten können.

Die Kommunikation der einzelnen Föderationsteilnehmer untereinander regelt eine *Broker-Komponente*, die jeder der Teilnehmer implementieren muss. Dieser *Broker* hat zwei zentrale Aufgaben:

1. *Kommunikation:* Der *Broker* nimmt zum einen die *Service Requests* der Kunden entgegen und leitet sie an den internen *SLA-Manager* weiter. Dieser verarbeitet die *Service Requests* und schließt entsprechende *SLAs* mit den Kunden. Zum anderen

ist der *Broker* für die Kommunikation der Föderationsteilnehmer untereinander zuständig. Dazu kann er mit anderen *RZs* des selben *CSPs*, mit anderen *CSPs* oder *vCSPs* sprechen.

2. *VM-Scheduling:* Der *Broker* besitzt weiterhin einen *VM-Scheduler*, der in 5.3 genauer beschrieben wird. Er ist für das *SLA-basierte Scheduling* der *VMs* im eigenen *RZ* (bei *CSPs*) zuständig, kann aber *VMs* auch in andere *RZs* des selben *CSPs* (intra *CSP*) oder in RZs anderer *CSPs* (inter *CSP*) migrieren (*CSPs* und *vCSPs*). Hierbei wird vorausgesetzt, dass alle Föderationspartner die gleichen technologischen Standards einsetzten.

Ein weiteres Element des Föderationsmodells ist der Kunde. Kunden stellen *Service Requests* an die *CSPs* oder *vCSPs*. Ein Kunde kann potentiell an alle Föderationspartner Anfragen stellen. *Service Requests* werden bei einer positiven Beantwortung in ein gültiges *SLA* umgewandelt und die entsprechenden *VMs* werden für den Kunden gestartet. Wie dieser Vorgang genau abläuft, wird unter 5.2 beschrieben.

Mit Hilfe der hier vorgestellten Komponenten lassen sich beliebige, unterschiedlich komplexe Föderationsumgebungen abbilden. Dadurch, dass jede Föderationskomponente einen eigenen *Broker* mit einem *Scheduler* beinhaltet, kann mit Hilfe dieses Modells ein verteilter *Cloud-Scheduling-Ansatz* untersucht werden. Durch die Integration der *vCSPs* können dabei zum einen neue Dienstleistungsangebote in verteilten Umgebungen untersucht werden, zum anderen kann aber auch ein zentraler Verwaltungsansatz nachgebildet werden. Dies gelingt durch den Einsatz eines zentralen *vCSPs* der ausnahmslos alle *Service Requests* aller Kunden entgegen nimmt und bearbeitet. Durch diesen Ansatz wird eine hohe Flexibilität bei der Erstellung von unterschiedlichen *Cloud-Szenarien* erreicht.

5.2 Föderierte Cloud-SLAs

Innerhalb des föderierten *Cloud-Modells* wird zwischen zwei verschiedenen Arten von *SLAs* unterschieden. Zum einen gibt es die *SLAs*, die die *(v)CSPs* untereinander schließen, um die Föderation aufzubauen. Zum anderen gibt es die *SLAs* zwischen den Kunden und den *(v)CSPs*. Zur Vereinfachung wird im weiteren Verlauf dieses Berichtes nicht weiter zwischen diesen beiden Arten unterschieden, da sie vom Aufbau her identisch sind. Die beiden *SLA-Typen* werden im Folgenden näher beschrieben:

1. *SLAs zwischen (v)CSPs:* Um initial eine Föderation zu gründen müssen mindestens zwei *CSPs SLAs* untereinander vereinbaren. Darin werden alle, für den jeweils anderen zur Verfügung gestellten *Services*, deren *QoS-Parameter* und die jeweiligen Kosten festgehalten. Zur Laufzeit können weitere *CSPs* hinzukommen und ihre *Services* in Teilen oder komplett in der Föderation zur Verfügung stellen. Wichtig zu beachten ist, dass jeder *CSP* mit jedem weiteren *CSP* eigene *SLAs* abschließen muss. Somit können gleiche *Services* mit unterschiedlichen *QoS-Parametern* und Kosten an unterschiedliche Föderationspartner angeboten werden. *vCSPs* schließen auch *SLAs* mit den jeweiligen *CSPs* ab, ohne aber eigene *Services* anzubieten. Die

Services aus den so vereinbarten *SLAs* können später den Kunden angeboten werden. Weiterhin gilt es zu beachten, dass die *SLAs* zwischen den *(v)CSPs* abstrakt die Ausführungskriterien der angebotenen *Services* beschreiben. Diese *SLAs* sind somit als eine Art Schablone zu verstehen, die die Dienstgütekriterien eines später, durch einen Kunden angefragten *Service*, beschreibt, welche sich die *CSPs* untereinander zusichern. *SLAs*, die sich auf die Ausführung eines konkreten *Services* beziehen, werden im nächsten Punkt erklärt.

2. *SLAs zwischen Kunden und (v)CSPs:* Kunden können zur Laufzeit *Services* bei einem *(v)CSP* bestellen. Dazu schicken sie einen *Service Request* an einen *CSP* oder einen *(v)CSP*. Der *SLA-Manager* des entsprechenden *RZs*, bei dem der *Service Request* eingereicht wurde, schließt dann ein *SLA* mit dem Kunden. Das der *Service* im Hintergrund vielleicht gar nicht in dem *RZ* ausgeführt wird, über den der *Service Request* eingereicht wurde, ist für den Kunden vollständig transparent. Dies ist ein großer Vorteil der *Cloud-Föderation*.

Die Integration von föderierten *SLAs* in das bisherige Modell erfordert eine weitere Komponente, den *SLA-Manager*. Der *SLA-Manager* übernimmt vier zentrale Aufgaben:

1. *SLAs abschließen:* Wenn ein Kunde die Ausführung eines *Services* anfragt, schickt er einen *Service Request*. Dieser *Service Request* wird vom *Broker* des entsprechenden *RZs* entgegen genommen und an den *SLA-Manager* durchgereicht. Dieser prüft zuerst, ob der *Service* lokal angeboten wird. Dazu schaut der *SLA-Manager* in einer Liste nach, die alle *Services* der Föderation und die des jeweiligen *RZs* beinhaltet. Wird der *Service* lokal angeboten, muss der lokale *Scheduler* entscheiden, ob die entsprechende *VM* auf einem der *Server* gestartet werden soll. Wird diese Anfrage positiv beantwortet, so generiert der *SLA-Manager* aus dem *Service Request* ein *SLA*, das die angefragten *Service-Parameter* beinhaltet. Das *SLA* wird dann in einer lokalen Datenbank abgelegt und die *VM* kann gestartet werden.

2. *SLAs überwachen:* Zur Laufzeit müssen alle aktiven *SLAs* überwacht werden. Dazu prüft der *SLA-Manager* periodisch mit Hilfe der Werte des *Monitorings* alle *QoS-Parameter* der *SLAs*. Werden alle *QoS-Werte* eingehalten, ist keine weitere Maßnahme erforderlich. Droht ein *SLA* zu reißen oder ist bereits gerissen, alarmiert der *SLA-Manager* den *Broker*. Dieser kann dann entsprechende Aktionen durchführen, damit das *SLA* schnellstmöglich wieder eingehalten wird.

3. *SLAs aufbrechen:* Besteht ein *Service* aus mehreren *VMs*, so kann der *Broker* entscheiden, dass die *VMs* in verschiedenen *RZs* ausgeführt werden sollen. Dazu muss dies natürlich möglich sein. Manche *Services* erfordern die Ausführung aller zugehörigen *VMs* in einem *RZ* oder sogar auf einem physikalischen *Server*. Dies kann über entsprechende *QoS-Parameter* ausgedrückt werden. Ist eine Aufteilung auf verschiedene *RZs* möglich, so muss natürlich der Teil, welcher diese *VM* beschreibt, aus dem *SLA* herausgetrennt werden. Der *SLA-Manager* erzeugt dann einen neuen *Service Request* für diesen Teilservice, welcher über den *Broker* an ein weiteres *RZ* durchgereicht werden kann.

4. *SLAs beenden:* Wird ein *Service* beendet, so muss natürlich auch das entsprechende *SLA* beendet werden. Der *SLA-Manager* entfernt dazu das *SLA* aus der Liste der aktiven *SLAs* und archiviert den Vertrag.

5.3 Template für einen SLA-basierten, Multi-Level-VM-Scheduler

Da der Bereich des *SLA-basierten VM-Schedulings* in *Cloud-Föderationen* bisher wenig erforscht ist, wird im Folgenden ein allgemeines *Template* für einen *SLA-basierten Multi-Level-Scheduler* beschrieben, welches es ermöglicht, anhand des oben beschriebenen Föderationsmodells, verschiedenste Experimente durchzuführen. Der bereits unter 5.1 beschriebene *Broker* beinhaltet diesen *VM-Scheduler*.

Das hier beschriebene *Template* ist in der Lage den gesamten Lebenszyklus einer *VM* abzubilden. Dazu muss beim *Scheduling* zwischen zwei Arten unterschieden werden:

1. *Initiales Schedule:* Hierbei handelt es sich um die initiale Platzierung einer neu erzeugten *VM* vom auf einem *Server*. Der entsprechende *Scheduling-Prozess* wird in Abbildung 1 grafisch dargestellt.
2. *Live-Schedule:* Hierbei handelt es sich um die *Live-Migration* einer *VM* zur Laufzeit. Der entsprechende *Scheduling-Prozess* wird in Abbildung 2 grafisch dargestellt.

Bei dem hier beschriebenen *Scheduler* handelt es sich um einen *SLA-basierten Multi-Level-VM-Scheduler*, der auf drei verschiedenen Ebenen arbeitet:

Ebene 1: Lokal Der *Scheduler* kann auf der lokalen Ebene (dem eigenen *RZ*) entscheiden, auf welchem physikalischen *Server* eine *VM* ausgeführt werden soll.

Ebene 2: Intra-CSP Der *Scheduler* kann auf der *Intra-CSP-Ebene* entscheiden, in welches *RZ* des selben *CSPs* eine *VM* migriert werden soll.

Ebene 3: Inter-CSP Der *Scheduler* kann auf der *Inter-CSP-Ebene* entscheiden, in welches föderierte *RZ* eines anderen *CSPs* eine *VM* migriert werden soll.

Dadurch, dass jeder Föderationsteilnehmer den hier beschriebenen *Broker* implementieren muss, lassen sich jetzt verteilte Verwaltungsansätze untersuchen. Bei dem vorgestellten Modell handelt es sich um ein abstraktes *Template* zur Erzeugung von speziellen Experimentierumgebungen. Dies liegt darin begründet, dass keine speziellen Entscheidungsstrategien vorgeschrieben werden, sondern, dass an insgesamt vier Stellen im Modell des *Schedulers* eine Schnittstelle vorgesehen ist, welche es erlaubt, beliebige Entscheidungsstrategien zu implementieren (siehe Abbildungen 1 und 2). Somit können bereits erforschte Strategien, wie zum Beispiel energieeffiziente *Scheduler* in verteilten, *SLA-basierten* Umgebungen neu getestet werden. Genauso ist es natürlich möglich, neue, speziell an die Einhaltung von *SLAs* angepasste *Scheduling-Strategien* zu implementieren und zu testen.

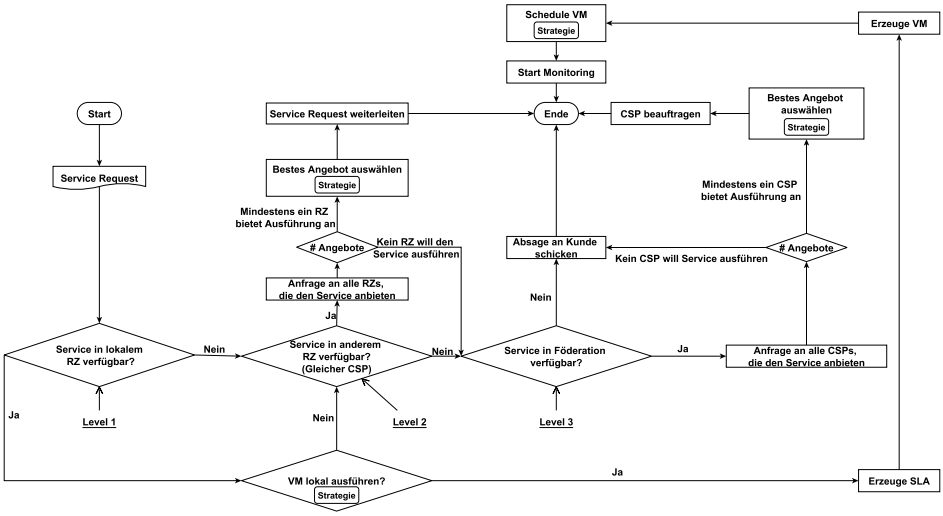

Abbildung 1: Grafische Darstellung des initialen *Multi-Level-Schedulings*

Auf den Ebenen 2 und 3 befindet sich jeweils ein *Interface*. Hier können entsprechende *Scheduling-Strategien* implementiert werden, die für die initiale *VM-Platzierung* oder die *Live-VM-Migration* in anderen *RZs* zuständig sind. Auf der ersten Ebene gibt es zwei Schnittstellen. Die erste kann dafür genutzt werden, eine Strategie zu implementieren, die entscheiden kann, ob neu von Kunden oder anderen *(v)CSPs* angefragte *VMs* überhaupt lokal ausgeführt werden sollen oder nicht. Mögliche Ziele der *Scheduling-Strategie* könnten hier sein: Energieeffizienz, Kostensteigerung, Gewinnoptimierung usw. Die zweite Schnittstelle auf Ebene 1 ist verantwortlich für den eigentlichen *VM-Scheduler*, welcher für die initiale Platzierung und *Live-Migration* der *VMs* auf den physikalischen *Servern* im lokalen *RZ* verantwortlich ist.

6 Diskussion

Der Einsatz von *Cloud-Föderationen* zur flexiblen Bereitstellung von *VMs* ist ein vielversprechender Ansatz, der in zahlreichen Projekten bereits untersucht wurde. Auch in der Wirtschaft wird sich dieser Ansatz langsam durchsetzten. Föderationen bieten für die *CSPs* wie auch für die Kunden viele Vorteile. Die *CSPs* können ihre lokalen Ressourcen effizienter einsetzen und bei Bedarf auf föderierte Ressourcen zurückgreifen. Für die Kunden potenziert sich das *Service-Angebot* der *CSPs* mit jedem neuen Föderationsmitglied. Auch der neu aufkommende Ansatz der *vCSPs* bietet viele interessante Ansätze. Durch die Aggregation verschiedenster *CPS-Services* über einen Marktplatz können ganz neue Geschäftsmodelle gestaltet werden.

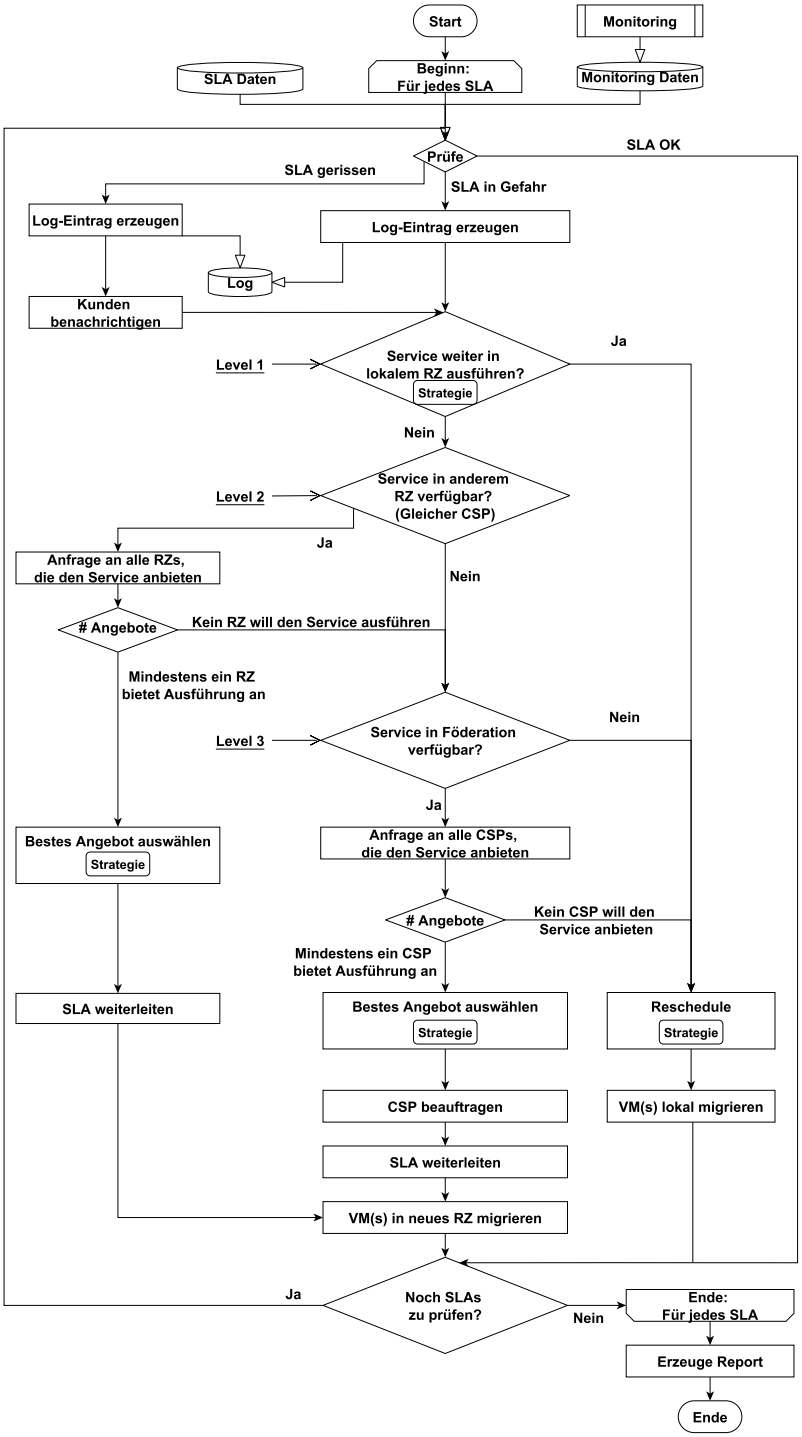

Abbildung 2: Grafische Darstellung des *SLA-basierten Multi-Level-Schedulings*

Bis sich *SLA-basierte Cloud-Föderationen* durchgesetzt haben, sind noch viele Fragen zu klären: Über welche Protokolle können Daten sicher zwischen *CSPs* ausgetauscht werden? Wie lassen sich *SLAs* flexibel und maschinenlesbar abbilden und vollautomatisch zwischen *(v)CSPs* und Kunden aushandeln? Wie kann garantiert werden, dass die *SLAs* in verteilten Szenarien eingehalten werden? Welche Auswirkungen hat dieser Ansatz auf die Datensicherheit? Wie wird mit Vertrauen in *Cloud-Föderationen* umgegangen?

Es gibt also noch viele Bereiche, in denen die Erstellung und Verwaltung von *Cloud-basierten Services* noch verbessert werden können. Einer dieser Bereiche ist das *SLA-basierte*, verteilte *Scheduling*. Insgesamt bietet das hier vorgestellte Föderationsmodell viele Forschungsmöglichkeiten und durch den Einsatz eines *Template-basierten Scheduling-Ansatzes* können verschiedenste *Cloud-Szenarien* und Entscheidungsstrategien untersucht werden.

7 Zusammenfassung und Ausblick

In diesem Beitrag wurde aufgezeigt, wie sich *Cloud-Föderationen* in Zukunft bilden können. Da es sich hierbei um einen relativ jungen Bereich der *Cloud-Forschung* handelt, ergeben sich noch viele interessante Aspekte. Ein wichtiger Aspekt ist dabei das *SLA-basierte VM-Scheduling* in verteilten *Cloud-Umgebungen*. Deshalb wurde hier ein Modell für ein *SLA-basiertes VM-Scheduling* in föderierten *Cloud-Umgebungen* beschrieben. Dazu wurden die einzelnen Komponenten der *Cloud-Föderation* mit ihren jeweiligen Eigenschaften im Detail beschrieben. Darauf aufbauend wurde ein *Template* für einen allgemeinen, *SLA-basierten Multi-Level-VM-Scheduler* vorgestellt.

Der große Vorteil dieses Ansatzes liegt darin, dass er es ermöglicht, beliebige Föderationsszenarien abzubilden und durch den Einsatz eines abstrakten *Multi-Level-Ansatzes* beliebige Entscheidungsstrategien auf allen drei Ebenen zu untersuchen. Ein weiterer Vorteil liegt darin, dass anhand dieses Modells nicht nur die Initial-Platzierung der *VMs* untersucht werden kann, sondern auch die *SLA-basierte Live-Migrationen* zur Laufzeit mit einbezogen wird. Somit lässt sich der gesamte Lebenszyklus einer *VM* untersuchen.

In einem nächsten Schritt wird das hier vorgestellte Modell mit Hilfe des *CloudSim Frameworks* implementiert. Hierzu sind einige Anpassungen und Erweiterungen am System nötig. Danach kann damit begonnen werden, bereits erforschte *Cloud-Scheduling-Strategien* sowie neue, *SLA-basierte* Strategien auf allen drei Ebenen des *Multi-Level-Schedulers* zu implementieren und zu testen. Dies soll anhand von realen *Cloud-Workload-Traces* zeigen, dass sich der Zusammenschluss zu einer *Cloud-Föderation* positiv auf alle Kooperationspartner auswirkt.

Literatur

[Ber07] G. Berger, Thomas. *Service-Level-Agreements : Konzeption und Management von Service-Level-Agreements für IT-Dienstleistungen*. Saarbrücken : VDM, Müller, 2007.

[BRC10] Rajkumar Buyya, Rajiv Ranjan und Rodrigo Calheiros. Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services. *Algorithms and architectures for parallel processing*, Seiten 13–31, 2010.

[BVMB08] Ivona Brandic, Srikumar Venugopal, Michael Mattess und Rajkumar Buyya. Towards a meta-negotiation architecture for SLA-Aware grid services. In *Workshop on Service-Oriented Engineering and Optimizations*, Seiten 17–20. Citeseer, 2008.

[CKR+10] Marco Comuzzi, Constantinos Kotsokalis, Christoph Rathfelder, Wolfgang Theilmann, Ulrich Winkler und Gabriele Zacco. A framework for multi-level SLA management. In *Service-Oriented Computing. ICSOC/ServiceWave 2009 Workshops*, Seiten 187–196. Springer, 2010.

[CRB+11] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, Cesar A. F. De Rose und Rajkumar Buyya. CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms. *Software: Practice and Experience (SPE)*, 41(1):23–50, 2011.

[CSM03] Adrian Ching, Lionel Sacks und Paul McKee. SLA management and resource modelling for grid computing. In *London Communications Symposium (LCS 2003), London, UK*, 2003.

[CTVP10] Antonio Celesti, Francesco Tusa, Massimo Villari und Antonio Puliafito. How to enhance cloud architectures to enable cross-federation. In *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, Seiten 337–345. IEEE, 2010.

[eas13] EASI CLOUDS, http://www.easi-clouds.eu, April 2013.

[idc11] IDC-Studie: Cloud Computing in Deutschland 2011, Juni 2011.

[JTS12a] Foued Jrad, Jie Tao und Achim Streit. Simulation-based Evaluation of an Intercloud Service Broker. In *CLOUD COMPUTING 2012, The Third International Conference on Cloud Computing, GRIDs, and Virtualization*, Seiten 140–145, 2012.

[JTS12b] Foued Jrad, Jie Tao und Achim Streit. SLA Based Service Brokering in Intercloud Environments, 2012.

[PRS09] Pankesh Patel, Ajith Ranabahu und Amit Sheth. Service Level Agreement in cloud computing. In *Cloud Workshops at OOPSLA*, 2009.

[SJD12] Jyothi Sekhar, Getzi Jeba und S Durga. A Survey on Energy Efficient Server Consolidation Through VM Live Migration. *International Journal of Advances in Engineering & Technology,*, 2012.

[Ten11] Fei Teng. *Management des donnees et ordonnancement des taches sur architectures distribuees*. Dissertation, Eole Centrale Paris, 2011.

[TP12] Sujit Tilak und Dipti Patil. A Survey of Various Scheduling Algorithms in Cloud Environment. *International Journal of Engineering Inventions*, 1(2):36–39, 2012.

[VT10] D. Versick und D. Tavangarian. Reducing Energy Consumption by Load Aggregation with an Optimized Dynamic Live Migration of Virtual Machines. In *P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2010 International Conference on*, Seiten 164–170, 2010.