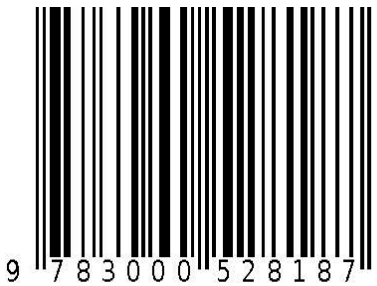
Hochschule Reutlingen
Reutlingen University

INF
Informatik

Uwe Kloos, Natividad Martínez, Gabriela Tullius (Hrsg.)

Informatics Inside
Digital Revolution

Informatik-Konferenz an der Hochschule Reutlingen
04. Mai 2016



Impressum

Anschrift:
Hochschule Reutlingen
Reutlingen University
Fakultät Informatik
Human-Centered Computing
Alteburgstraße 150
D-72762 Reutlingen

Telefon: +49 7121 / 271-4002
Telefax: +49 7121 / 271-4042

E-Mail: infoinside@reutlingen-university.de
Internet: http://www.infoinside.reutlingen-university.de

Organisationskomitee:
Prof. Dr. Gabriela Tullius, Hochschule Reutlingen
Prof. Dr. Natividad Martínez, Hochschule Reutlingen
Prof. Dr. Uwe Kloos, Hochschule Reutlingen

Palina Vorobeva
Julian Freund
Armando Statti
Nils Tofahrn
Thomas Walzer
Natascha Stumpp
Damir Stazic

Hochschule Reutlingen
Reutlingen University

Herstellung und Verlag: Hochschule Reutlingen
ISBN 978-3-00-052818-7

Inhaltsverzeichnis

Paper

Shortpaper

Evaluierung der Lastverteilung und Skalierung von Cloud-Plattformen*

Alexander Kunz
Reutlingen University
alexander.kunz@Student.
Reutlingen-University.DE
Reutlingen University

Abstract

Durch das breite Angebot an Cloud-Plattformen fällt es schwer, die passende Plattform für einen bestimmten Anwendungsfall auszuwählen. Es wird häufig die Frage gestellt, welche Unterschiede die einzelnen Cloud-Plattformen aufweisen und welche Eigenschaften und Vorteile jede einzelne besitzt. In diesem Artikel werden deshalb zunächst die Prinzipien von Cloud Computing näher gebracht. Außerdem werden die Plattformen Amazon Web Services, Microsoft Azure, Pivotal Cloud Foundry und OpenStack näher beleuchtet und auf die Aspekte der Skalierung und Lastverteilung untersucht.

Schlüsselwörter

Cloud Computing, Skalierung, Lastverteilung

CR-Kategorien

C.4 [**Performance of Systems**]: Performance attributes

---

*

Betreuer Hochschule: Prof. Dr. Marcus Schöller
Hochschule Reutlingen
Marcus.Schoeller@Reutlingen-University.de
Betreuer Firma: Johannes Dilli
Novatec Consulting GmbH
Johannes.Dilli@novatec-gmbh.de
Informatics Inside 2016
Wissenschaftliche Vertiefungskonferenz
4. Mai 2016, Hochschule Reutlingen

1 Einleitung

Cloud Computing ist eine Erweiterung von Grid Computing, Parallel Computing und Distributed Computing [10, 5]. Es hat in den letzten Jahren eine immer größere Bedeutung erhalten, wodurch ein großes Angebot an Plattformen entstanden ist. Dadurch fällt es schwer, die passende Cloud-Plattform für einen bestimmten Anwendungsfall auszuwählen. Mit Cloud Computing und dem einhergehenden pay-per-use-Bezahlmodell, sind Infrastruktur und Rechenleistung für Unternehmen schnell und günstig zu beziehen. Pay-per-use bedeutet, dass nur die Ressourcen bezahlt werden müssen, die verwendet wurden. Die Unternehmen müssen damit keine eigene Infrastruktur bereit stellen, sondern bekommen on-demand genau die Ressourcen von den Cloud-Anbietern zur Verfügung gestellt, die benötigt werden. Dadurch entsteht ein großes Einsparpotential für die Unternehmen. Neue Benutzer von Cloud Computing, stehen zu Beginn vor dem Problem, die passende Plattform für den speziellen Anwendungsfall auszuwählen. Deshalb sollen in dieser Arbeit verschiedene Cloud-Plattformen auf ihre Funktionsweise in der Lastverteilung und Skalierung evaluiert werden.

In Abschnitt 2 werden zunächst die Ziele der Arbeit beschrieben. Darauffolgend wird in Abschnitt 3 eine Einführung in Cloud Computing gegeben und anschließend, in Abschnitt 4 und 5, die verschiedenen Plattformen vorgestellt und evaluiert.

2 Ziele der Arbeit

In dieser Arbeit sollen mehrere Cloud-Plattformen betrachtet werden. Dazu werden im nächsten Abschnitt die Prinzipien von Cloud-Computing näher gebracht und erläutert, welche Probleme mit Cloud-Computing gelöst werden können und welche Vor- bzw. Nachteile eine Cloud-Lösung mit sich bringt. Weiterhin werden die technischen Aspekte der Plattformen von Amazon Web Services[1], Microsoft Azure[2], OpenStack[3] und Pivotal Cloud Foundry[4] genauer untersucht. Dabei sollen die Möglichkeiten, welche die Plattformen zur Lastverteilung und Skalierung anbieten, analysiert und die Unterschiede ermittelt werden. Es soll ermittelt werden, welche Vor- und Nachteile die Plattformen und Konzepte für verschiedene Anwendungen bieten.

3 Cloud Computing

Unter Cloud Computing versteht man die dynamische Verwendung von entfernten IT-Ressourcen. Die fundamentalen Eigenschaften von Cloud Computing sind Elastizität, pay-per-use, Standardisierung, Virtualisierung und eine gemeinsame Ressourcennutzung [2]. Die Elastizität ermöglicht den Benutzern, Ressourcen dynamisch zu reservieren und wieder freizugeben. Mithilfe des pay-per-use-Bezahlmodells wird sichergestellt, dass den Benutzer nur die Ressourcen in Rechnung gestellt werden, die sie verwenden. Die Virtualisierung ist ein weiterer wichtiger Bestandteil von Cloud-Plattformen. Hierbei werden mehrere virtuelle Rechner auf einem physikalischen ausgeführt. Dadurch entsteht eine sichere, skalierbare und überschaubare Umgebung [3, 5]. Die Skalierung und Lastverteilung spielt im Cloud-Umfeld ebenfalls eine wichtige Rolle. Mit Skalierung ist meistens eine automatische Skalierung gemeint, bei welcher bestimmte Regeln festgelegt werden können, um automatisch Ressourcen hinzuzufügen oder abzuschalten. Die Lastverteilung sorgt dafür, dass die Last (z.B. die Anfragen der Clients) gleichmäßig auf alle aktiven Instanzen verteilt wird. Bei Cloud Computing wird meist zwischen drei verschiedenen Modellen unterschieden: Infrastructure-as-a-Service (IaaS), Plattform-as-a-Service (PaaS) und Software-as-a-Service (SaaS).

[1] https://aws.amazon.com/
[2] https://azure.microsoft.com/en-us/
[3] https://www.openstack.org/
[4] http://pivotal.io/platform

3.1 Infrastructure-as-a-Service

Bei Infrastructure-as-a-Service verwaltet der Anbieter alle Hardwareaspekte des Cloud-Systems [4]. Der Anbieter stellt die Server und den Speicherplatz bereit und kümmert sich um die Instandhaltung, während zusätzlich die nötige Software zur Virtualisierung zur Verfügung gestellt wird. Mithilfe dieser, werden die physikalischen Server von den virtuellen getrennt, wodurch eine echte Cloud-Infrastruktur ermöglicht wird. IaaS stellt meist zusätzliche Dienste zur Verfügung, wie beispielsweise Lastverteilung oder Firewalls. Die Kunden, die IaaS nutzen, bekommen damit eine optimale Voraussetzung für das Produkt, das mithilfe der Infrastruktur betrieben werden soll, und haben freie Entscheidung im Hinblick auf Betriebssystem, Middleware und der Software.

3.2 Plattform-as-a-Service

Plattform-as-a-Service liegt zwischen IaaS und SaaS. Hier handelt es sich nicht, wie bei SaaS, um ein fertiges Produkt, das direkt vom Kunden benutzt werden kann. Es ist aber auch kein System wie IaaS, auf dem alles betrieben werden kann. Bei PaaS werden IaaS Ressourcen mit einem Betriebssystem, einer Middleware und Laufzeitumgebung ausgestattet [4]. Damit wird eine Plattform zur Verfügung gestellt, auf der Anwendungen betrieben werden können, ohne den Verwaltungsaufwand von niedrigeren Schichten berücksichtigen zu müssen. Der Nachteil bei PaaS ist, dass die Software für eine spezielle Plattform entwickelt wird, über die der Entwickler keine Kontrolle hat und damit eine Abhängigkeit zum Anbieter entsteht.

3.3 Software-as-a-Service

Mit Software-as-a-Service wird das Service-Modell bezeichnet, welches die meisten Internetbenutzer in Anspruch nehmen. Hierbei fallen alle Anwendungen, die auf einem entfernten Server betrieben werden und über ein Netzwerk erreicht werden können, unter die Kategorie SaaS [4]. In den meisten Fällen bekommen die Nutzer über einen Webbrowser zugriff auf eine Anwendung. Beispiele sind hier Google Mail[5] oder Netflix[6].

4 Cloud Plattformen

In diesem Abschnitt werden die Plattformen vorgestellt, die evaluiert werden sollen. Hierbei wurden vier Plattformen ausgewählt: Amazon Web Services, Microsoft Azure, Cloud Foundry und OpenStack.

4.1 Amazon Web Services

Amazon Web Services (AWS) bietet eine Vielzahl an Cloud-Diensten an. In dieser Arbeit wurde der Plattform-as-a-Service Dienst Elastic Beanstalk näher untersucht [1]. Der Dienst ermöglicht es dem Benutzer Anwendungen in der Amazon Cloud in Betrieb zu nehmen. Beanstalk kümmert sich dabei automatisch um die Beschaffung der Instanzen, Lastverteilung, Skalierung und Monitoring der Anwendung.

Elastic Load Balancing verteilt den eingehenden Verkehr automatisch auf verschiedene Elastic Compute Cloud (EC2) Instanzen. Dabei kann die eingehende Last ebenfalls auf verschiedene Verfügbarkeitszonen (Availability Zone) verteilt werden. Verfügbarkeitszonen sind Standorte, an denen die EC2 Instanzen von Amazon verwaltet werden. Zusätzlich ist der Elastic Load Balancer selbst ein verteiltes System, welches fehlertolerant ist und ebenfalls aktiv überwacht wird.

Weiterhin wird die Integration von Auto Scaling angeboten, wodurch in Echtzeit auf steigende oder sinkende Last reagiert werden kann. Beispielsweise kann die automatische Skalierung so konfiguriert werden, dass mindestens drei Instanzen aktiv sind. Wird die Last größer und es werden bestimmte Grenzen erreicht (wie z.B. CPU-Auslastung oder eine bestimmte Anzahl an Anfragen), werden automatisch weitere Instanzen hinzugefügt.

4.2 Microsoft Azure

Ebenso wie Amazon Web Services, bietet Microsoft Azure eine große Produktpalette im Cloud-Umfeld an. Hier wurde Azure Cloud Services näher betrachtet, welcher ebenfalls ein Plattform-as-a-Service Dienst ist [6]. Der Dienst übernimmt ebenso wie bei AWS die Beschaffung der Instanzen, Lastverteilung, Skalierung und Monitoring der Anwendung.

Anfragen an eine Azure-Anwendung, werden automatisch über den Load Balancer auf die vorhanden Instanzen verteilt. Der Load Balancer ordnet die öffentliche IP-Adresse der eingehenden Anfragen den privaten IP-Adressen der einzelnen Instanzen zu. Für die automatische Skalierung kann in der Konfiguration ähnlich wie bei AWS eingestellt werden, wann neue Instanzen hinzugefügt oder abgeschaltet werden sollen. Bei Cloud Service Anwendungen kann jedoch zunächst nur auf die CPU-Auslastung reagiert werden und nicht etwa wie bei AWS auf die Anzahl der Anfragen.

4.3 OpenStack

OpenStack hat das Ziel, eine Open Source Cloud Computing Plattform für private und öffentliche Clouds zu erstellen. Der Fokus von OpenStack liegt dabei auf der Skalierbarkeit der Anwendungen und Infrastructure-as-a-Service [7].

Der Load Balancer von OpenStack ermöglicht es, die eingehenden Anfragen gleichmäßig auf die vorhandenen Instanzen zu verteilen. Dabei kann zwischen drei Methoden ausgewählt werden: Eingehende An-

[5] https://mail.google.com/
[6] https://www.netflix.com/de-en/

fragen werden im Round Robin Prinzip gleichmäßig an die Instanzen geleitet, Anfragen der gleichen IP-Adresse werden immer an die selbe Instanz geleitet oder die Anfragen werden an die Instanz geleitet, welche die wenigsten aktiven Verbindungen besitzt. Aufgrund dessen, dass OpenStack ein Infrastructure-as-a-Service Projekt ist, ist der Aufwand eine Anwendung in einer OpenStack Cloud in Betrieb zu nehmen und mit einem Load Balancer zu skalieren vergleichsweise hoch. Ein Autoscaling und Load Balancer kann mithilfe der Projekte Heat[7] und Ceilometer[8] entsprechend konfiguriert werden.

4.4 Cloud Foundry

Cloud Foundry ist, ebenso wie OpenStack, ein Open Source Projekt und besteht aus vielen Komponenten und Diensten, um Anwendungen zu verbreiten, skalieren und überwachen. Im Gegensatz zu OpenStack wird Cloud Foundry jedoch in die Kategorie Plattform-as-a-Service eingeordnet [8].

Für die Lastverteilung setzt Cloud Foundry einen HAProxy[9] Load Balancer ein und bietet ebenfalls (wie AWS) die Möglichkeit an, unterschiedliche Verfügbarkeitszonen einzurichten. Diese besitzen eine ähnliche Funktionsweise wie die Verfügbarkeitszonen von AWS. Eine Anwendung wird beispielsweise auf drei Verfügbarkeitszonen (AZ1, AZ2 und AZ3) und insgesamt vier Instanzen verbreitet. Die Zone AZ1 hält zwei Instanzen, AZ2 und AZ3 jeweils eine Instanz. Fällt AZ1 aus, wird der eingehende Verkehr nur noch auf AZ2 und AZ3 geleitet. Sollte AZ1 nach einer bestimmten Zeit noch immer nicht erreichbar sein, werden auf AZ2 und AZ3 jeweils eine zusätzliche Instanz gestartet, damit die Anwendung insgesamt wieder auf 4 verschiedenen Instanzen läuft.

[7] https://wiki.openstack.org/wiki/Heat
[8] https://wiki.openstack.org/wiki/Telemetry
[9] http://www.haproxy.org/

5 Evaluation

In der Evaluation der Cloud-Plattformen wurde untersucht, wie sich die Plattformen mit mehreren Instanzen verhalten und ob sie so skalieren, wie es in der Konfiguration vorgegeben wurde. Weiterhin wurde geprüft, ob die Last gleichmäßig auf alle Instanzen verteilt wird.

5.1 Durchführung

Für die Durchführung der Evaluation mussten zunächst Kriterien festgelegt werden, damit die Plattformen miteinander verglichen werden können. Da die Skalierung und Lastverteilung untersucht werden sollten, wurden außerdem folgende Regeln zur Skalierung festgelegt: Beträgt die Gesamtauslastung der CPU über 70% soll horizontal nach oben skaliert werden. Es sollen jedoch nur maximal drei Instanzen in Betrieb genommen werden. Weiterhin soll erst dann nach oben skaliert werden, wenn die Auslastung über einen Zeitraum von mindestens fünf Minuten über 70% liegt. Liegt die Auslastung für 20 Minuten unter 50%, soll nach unten skaliert werden.

Da die Tests auf allen Plattformen mit freien Testaccounts durchgeführt wurden, sind gewisse Einschränkungen entstanden. Die Testumgebung TryStack[10] (OpenStack Cluster) limitiert die Nutzung beispielsweise auf maximal drei Instanzen. Um CPU-Last auf den Instanzen zu erzeugen, wurde Java SciMark 2.0 verwendet [9]. SciMark 2.0 ist ein Java Benchmark für wissenschaftliche und numerische Berechnungen. Der Benchmark führt mehrere komplexe Berechnungen durch wie beispielsweise eine Fast Fourier Transformation oder Monte Carlo Integration und gibt ein zusammengesetztes Ergebnis in Form von Mega Floating Point Operations Per Second (MFLOPS) aus. Im Folgenden, eine Auflistung der Kriterien, auf welche bei den Testdurchläufen geachtet wurden:

- Welche Instanzen verwendet die jeweilige Plattform?

[10] http://trystack.org/

- CPU-Auslastung einer Instanz bei laufendem Benchmark?
- Funktioniert das Scale-Up Event korrekt?
- Funktioniert das Scale-Down Event korrekt?
- Wird die Last gleichmäßig verteilt?
- Ergebnis des Benchmarks in MFLOPS?
- Wie groß war der Aufwand, um die Plattformen entsprechend zu konfigurieren und die Anwendung zu starten?

Um die Kriterien zu überprüfen, wurde eine Webanwendung erstellt, welche beim Aufruf den Benchmark ausführt und das Ergebnis auf einer Webseite darstellt. Die Webseite wurde dabei so entwickelt, dass mit jedem erhaltenen Ergebnis (also nach einem fertigen Durchlauf des Benchmarks) eine weitere Anfrage an die Anwendung gesendet wird, welche den Benchmark erneut ausführt.

5.2 Überblick

In Tabelle 1 ist zunächst eine Übersicht der Ergebnisse zu sehen. Die CPU-Auslastung repräsentiert dabei die CPU-Auslastung von einer Instanz bei laufendem Benchmark. Scale-Up und Scale-Down sagt aus, ob die Skalierung so durchgeführt wird, wie es in der Konfiguration angegeben wurde. Die Zeile der Lastverteilung gibt an, ob die Last gleichmäßig oder ungleichmäßig auf die Instanzen verteilt wurde. Außerdem ist das Ergebnis des Benchmarks auf einer Instanz der jeweiligen Plattform in der Tabelle zu sehen sowie eine Bewertung des Aufwands, eine Anwendung zu deployen.

Die CPU war nur bei Azure nicht vollkommen ausgelastet. Das Scale-Up funktionierte bei AWS und Azure so, wie es in der Konfiguration vorgegeben wurde. Bei OpenStack (bzw. TryStack) ist die Umsetzung mit großem Aufwand verbunden, da es ein Infrastructure-as-a-Service Projekt ist. Aufgrund dessen konnten die Tests zur Skalierung und Lastverteilung bei OpenStack nicht durchgeführt werden. Bei Pivotal Cloud Foundry konnte die Lastverteilung und Skalierung zwar konfiguriert werden, jedoch sind die Möglichkeiten sehr begrenzt. Es besteht beispielsweise keine einfache Möglichkeit ein Zeitlimit für die Skalierung anzugeben. Daher wird nach oben und unten skaliert, sobald die CPU-Auslastung die Grenze nur kurzzeitig über- bzw. unterschreitet. Die Lastverteilung mit den Standard Load Balancern der Plattformen sah ebenfalls nur bei Azure gleichmäßig aus. Bei AWS und Cloud Foundry hingegen, wurde die Last nicht immer erwartungsgemäß verteilt. Die Werte des Benchmarks können nur schwer erklärt werden, da es nicht ersichtlich ist, mit welchen Ressourcen die Instanzen der verschiedenen Plattformen arbeiten.

5.3 Skalierung und Lastverteilung

Um die Lastverteilung und Skalierung zu prüfen, wurde folgender Testlauf definiert: Es wird mit einer Instanz gestartet, welche den Benchmark dauerhaft ausführt (siehe 5.1). Bei zwei laufenden Instanzen wird die Last verdoppelt und bei drei Instanzen verdreifacht. In Abbildung 1 ist die Gesamtauslastung von AWS, Azure und Cloud Foundry für einen solchen Testdurchlauf zu sehen. Bei Minute 0 des Diagramms wird zunächst eine Instanz gestartet und mithilfe des Benchmarks CPU-Last erzeugt. Dabei tritt bei AWS und Azure nach 10 und 15 Minuten das Scale-Up Event auf, bei dem eine Instanz hinzugefügt wird. Damit verhalten sich die zwei Plattformen so, wie es in der Konfiguration angegeben wurde. Da bei Cloud Foundry kein Zeitlimit angegeben werden konnte, wurden hier alle Instanzen nach dem Muster von AWS und Azure manuell hinzugefügt bzw. abgeschaltet. Aufgrund dessen ist der Datensatz im Diagramm in gepunkteter Form abgebildet.

Tabelle 1: Ergebnis der Evaluierung

	AWS	Azure	TryStack	Cloud Foundry
Instanzen	Linux	Windows	Linux	Linux
CPU-Auslastung	100%	82%	100%	100%
Scale-Up	✓	✓	-	✓
Scale-Down	✓	✓	-	✓
Lastverteilung	ungleichmäßig	gleichmäßig	-	ungleichmäßig
SciMark	623	674	437	1319
Aufwand	niedrig	niedrig	hoch	niedrig

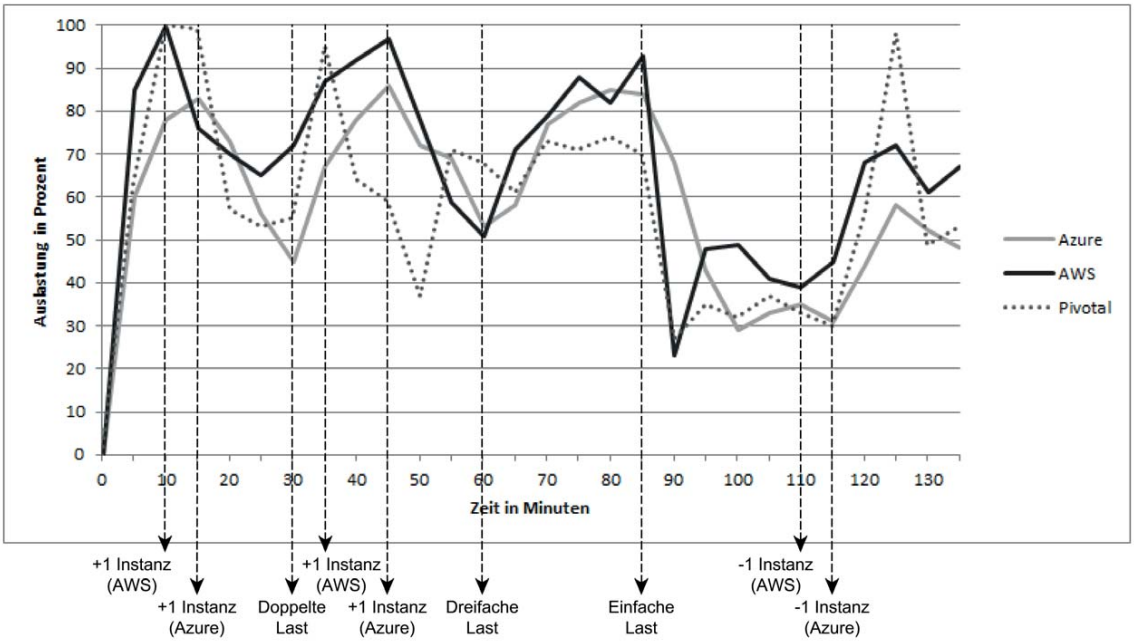


Abbildung 1: CPU-Gesamtauslastung von AWS, Azure und Pivotal Cloud Foundry

Die Gesamtauslastung der CPU kann mithilfe der zweiten Instanz bei allen Plattformen gesenkt werden. Bei Minute 30 wird ein weiterer Benchmark gestartet, womit die Auslastung bei jeder Plattform wieder steigt. Hierbei kann ein Unterschied in der Verhaltensweise der Plattformen erkannt werden: Bei Cloud Foundry wurde die Last nicht immer korrekt auf die zwei Instanzen verteilt, was durch eine geringere Gesamtauslastung ab Minute 35 erkennbar ist. Würde jeweils ein Benchmark auf einer Instanz ausgeführt werden, wäre eine Gesamtauslastung zwischen 90% und 100% zu erwarten. Es wird jedoch oftmals nur eine Instanz verwendet, wodurch eine geringere Gesamtauslastung entsteht. Die gleiche Problematik konnte ebenfalls bei AWS beobachtet werden, tritt dort aber nur in seltenen Fällen auf.

Da die Last in den vergangenen fünf Minuten oberhalb der 70% Marke liegt, wird bei Minute 35 und 45 wird eine weitere Instanz hinzugefügt. Ab diesem Zeitpunkt kann wieder beobachtet werden, wie die Gesamtauslastung zurückgeht, bis bei Minute 60 die dreifache Last erzeugt wird und die Auslastung erneut steigt. Da maximal drei Instanzen betrieben werden sollen, wird keine keine weitere Instanz mehr hinzugefügt und bei Minute 85 die Last wieder verringert. Wie oben beschrieben, wird nach unten skaliert, sobald die Gesamtauslastung über einen Zeitraum von 20 Minuten unter 50% liegt.

6 Fazit

Das Ergebnis der Evaluation zeigt, dass sich die Plattformen im Hinblick auf die Skalierung so verhalten, wie es vom Benutzer vorgegeben wird. Bei der Lastverteilung wurde hingegen nicht immer das erwünschte Ergebnis beobachtet. Einzig Microsoft Azure hat in den durchgeführten Tests die Last stets korrekt auf die vorhandenen Instanzen aufgeteilt. Bei AWS und Pivotal Cloud Foundry konnte beobachtet werden, dass Instanzen keine Last zugeteilt wurde. Dieses verhalten trat bei AWS eher seltener auf, bei Pivotal hingegen häufiger. Der Grund für die falsche Lastverteilung könnte daher stammen, dass Cloud Foundry keine normalisierte CPU-Auslastung zur Verfügung stellt und die Ressourcenbeschaffung nicht grundsätzlich auf diese Metrik ausgelegt ist. Bei IBM Bluemix, welche ebenfalls Cloud Foundry verwenden, wird die CPU-Auslastung für das Autoscaling beispielsweise nicht angeboten, da es nicht zuverlässig ist. Der Aufwand, um eine Anwendung in Betrieb zu nehmen und die Einarbeitung in die einzelnen Plattformen, kann für AWS, Azure und Cloud Foundry etwa gleich bewertet werden. Hier ist der Einstieg sehr gut Dokumentiert und anfängerfreundlich gehalten. Bei OpenStack hingegen ist etwas mehr Aufwand nötig (aufgrund von Infrastructure-as-a-Service), weshalb für die Plattform letztendlich kein Test zur Skalierung und Lastverteilung durchgeführt werden konnte.

Literatur

[1] Amazon. Amazon web services - elastic beanstalk documentation. Website. Online Verfügbar unter: `http://aws.amazon.com/documentation/elastic-beanstalk/` letzter Zugriff am 07.04.2016.

[2] C. Fehling, T. Ewald, F. Leymann, M. Pauly, J. Rutschlin, and D. Schumm. Capturing Cloud Computing Knowledge and Experience in Patterns. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pages 726–733, June 2012.

[3] C. Fehling, F. Leymann, R. Mietzner, and W. Schupeck. A Collection of Patterns for Cloud Types, Cloud Service Models, and Cloud-based Application Architectures. In *Technical Report No. 2011/05*, University of Stuttgart: Stuttgart, Germany, 2011.

[4] C. Fehling, F. Leymann, R. Retter, W. Schupeck, and P. Arbitter. *Cloud Computing Patterns - Fundamentals to Design, Build, and Manage Cloud Applications*. Springer, 2014.

[5] K. Hashizume, N. Yoshioka, and E. Fernandez. Misuse Patterns for Cloud Computing. In *Proceedings of the Asian Conference on Pattern Languages of Programs (AsianPLoP)*, Tokoy, Japan, 17–19 March 2011.

[6] Microsoft. Should I Choose Cloud Services? Website. Online Verfügbar unter: `https://azure.microsoft.com/en-us/documentation/articles/fundamentals-introduction-to-a` letzter Zugriff am 22.01.2016.

[7] K. Pepple. *Deploying OpenStack*. O'Reilly Media, Inc, 2011.

[8] Pivotal Cloud Foundry. Documentation. Website. Online Verfügbar unter: `http://docs.run.pivotal.io/concepts/` letzter Zugriff am 07.04.2016.

[9] P. Roldan and B. R. Miller. SciMark. Website. Online Verfügbar unter: `http://math.nist.gov/scimark2/` letzter Zugriff am 07.04.2016.

[10] S. Zhang, S. Zhang, X. Chen, and X. Huo. Cloud Computing Research and Development Trend. In *Future Networks, 2010. ICFN '10. Second International Conference on*, pages 93–97, Jan 2010.