Cloud Computing - Appscale (WS0910)

Florian Weispfenning

Fakultät für Informatik,
Hochschule Mannheim,
Paul-Wittsack-Straße 10,
68163 Mannheim
florian.weispfenning@stud.hs-mannheim.de

Veröffentlichung am 31.12.2009

Zusammenfassung Appscale ist ein Projekt, mit welchem man Anwendungen für Googles AppEngine auf eigenen Servern ausführen kann.

1 Motivation

Cloud Computing bietet die Möglichkeit, die Hardware von Servern effektiver nutzen zu können. Die Software wird auf Servern eines Cloud-Anbieters ausgeführt, welche von diesem verwaltet und gewartet werden. Die momentan im Internet stehenden Cloud-Server werden meinst von Betreibern im Ausland bereit gestellt. Genau hier können rechtliche Probleme entstehen oder wenn zum Beispiel sensible Daten verarbeitet sollen. Diese Daten dürfen nicht die eigene Firma verlassen und können somit nicht im Internet verarbeitet werden. Um Vorteile einer Cloud zu nutzten, muss die Cloud in die Firma kommen. Die Lösung ist eine private Cloud. Dieser Artikel soll den Aufbau solch einer lokalen cloudbasierenden Anwendungs-Platform am Beispiel von Appscale demonstrieren.

Momentan gibt es mit Microsoft Azure und Google's AppEngine zwei große *platform-as-a-service (PaaS)*-Anbieter. Mit AppScale ist man kompatibel zu Googles-AppEngine, hat aber den Vorteil nicht abhängig von Google zu sein, sodass man zur Not in eine eigene (oder von Drittanbietern) bereitgestellte Umgebung umziehen könnte. AppScale bietet auch einen Authentifizierungs-Mechanisums. Dieser basiert im Gegensatz zu Google nicht auf den Google-Accounts und kann komplett ohne Internetanbindung verwendet werden. Es wurde weiterhin eine Schnittstelle zu einer Menge an Datenbanken bereitgestellt. Zu diesen zählen unter anderem HBase, Voldemort oder MySQL. Da es sich bei AppScale um ein komplett transparentes System handelt, kann man an solch einer Installation das Verhalten einer Anwendung in der Cloud studieren, sodass dies ein Anreiz für Studium und Forschung sein könnte solch eine Installation aufzusetzten.

2 AppScale: Open Source PaaS

AppScale ist eine Plattform, mit welcher man Anwendungen für Google's AppEngine auf eigenen Servern ausführen lassen kann. Google's AppEngine und auch AppScale sind eine *platform-as-a-service (PaaS)*, welche Entwicklern die Möglichkeit bietet Anwendungen auszuführen zu lassen. Wie bei Google lassen sich die Anwendungen mit Java oder Python erstellen. Es ist als Forschungsprojekt [1] der University of California, Santa Barbara (UCSB) ins Leben gerufen worden. Das Kernstück des Projekts sind vorgefertigte Abbilder von virtuellen Maschinen für XEN [6] oder KVM [7]. Hit Hilfe eines mitgeliefert Administrations-Werkzeug für die Kommandozeile kann die komplette Anwendungs-Cloud verwaltet werden.

Das Projekt [3] ist am 7.3.2009 in der Version 1.0 veröffentlicht worden. Mittlerweile wurde am 14. Dezember diesen Jahres die Version 1.3 über Google-Code [4] zum Download angeboten. Auf dieser Projektseite ist neben dem Quelltext des Projektes eine Online-Community [5] zu finden, welche einem bei Problemen unter die Arme greift.

2.1 Hard- und Software Voraussetzungen

Die oben erwähnten Abbilder basieren auf Ubuntu Server 64Bit und setzten darum auch ein 64Bit Wirt-System voraus. Alternativ kann AppScale auch direkt in einem Linux-System installiert werden. Die Entwickler empfehlen auf Ubuntu Jaunty zu setzten, da auch die virtuellen Maschinen (VM) diese Version verwenden. Um das System aufbauen zu können benöigt man in irgend einer Form Hardware. Diese kann entweder über Xen oder KVM virtualisiert werden, oder direkt benutzt werden. Es können nur System zur Virtualisierung genuzt werden, welche Xen oder KVM unterstützten, sodass Windows nicht in Frage kommt. Für eine Appscale-Installation werden mehre Instanzen benötigt. Jede AppScale-VM stellt solch eine Instanz dar. Diese Instanz benötigt zwischen 0,5 und 4GB RAM und sollte, wenn möglich, zwei Kerne zu Verfügung gestellt bekommen. Der benötigte Festplattenspeicherplatz hält sich mit 6GB sehr in Grenzen. Für die Virtualisierung kann auf EC2 [8] oder Eucalyptus [9] zurück gegriffen werden.

3 Komponenten in einer AppScale-Installation

Appscale besteht aus mehreren Komponenten (siehe Abbildung 1), welche jede für sich eine spezielle Aufgabe übernimmt. Die ausgelieferte AppScale-VM beinhät alle Variationen und kann die benötigte Komponenten starten. Nachdem Start der VMs muss die „Cloud“ konfiguriert werden, indem man festlegt welcher Server für welche Aufgaben zuständig ist.

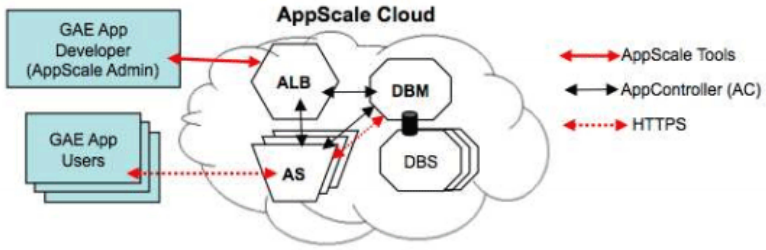

Abbildung 1. Übersicht der verfügbaren Komponenten

3.1 AppController (AC)

Der AppController verwaltet und kennt alle in der Cloud beteiligten Server. Nur über diesen Server können neue Anwendungen in das System eingespielt werden. Das Hochladen von neuen Anwendungen in die Cloud geschieht über die mitgelieferten Administrationswerkzeuge. Der AppController wurde in Ruby geschrieben. Damit die Anwendung auch auf Datenbanken zugreifen kann, sorgt der AppController dafür, dass der DMB gestartet wird. Eine neue Andwendung wird durch den AC auf die weiteren in der Cloud beteiligten Server (AS) kopiert. Falls im Laufe des Betriebes ein Server ausfällt, kann der AC diesen wieder starten. Alle Server der Cloud werden durch den AC im Zehn-Sekundentakt überprüft, dem sogenannte „heartbeat“. Bei diesem „heartbeat“ wird die RAM und CPU-Last abgefragt. Diese Werte werden für den AppLoadBalancer (ALB) zur Verfügung gestellt, welcher auch auf dem selben Server läuft.

3.2 AppLoadBalancer (ALB)

Auch der AppLoadBalancer (ALB) kennt alle in der Cloud beteiligten Server und sitzt darum auf dem gleichen System wie der AppController. Diese Komponente ist als eine Ruby on Rails [10] - Anwendung realisiert. Als Webserver wurde nginx [11] genutzt, da dieser sehr effizient arbeitet. Bei großen Webseiten, wie zum Beispiel `github.com`, `sourceforge.net` oder `Golem.de` wird auch auf nginx gesezt.

Mit dem ALB werden die Logins der Anwender verwaltet, sodass diese Komponente alle anzumeldenden Benutzer kennt. Wenn ein Anwender sich mit der Cloud zum ersten Mal verbindet, geschieht dies über den ALB. Mit Hilfe des „heartbeat“ kennt er alle die Auslastung aller in der Cloud verfügbaren Server (Kapitel 3.4) und leitet den Anwender zu dem Server mit der geringsten Auslastung. Falls mehrere Server ähnlich wenig Auslastung haben, wird der Anwender zufällig zu einem dieser Server zugewiesen. Nachdem der Anwender zu einem

Server zugewissen wurde, spielt sich die Kommunikation komplett zwischen diesem Server und dem Anwender ab. Durch diesen Aufbau entsteht eine starke Entlastung des ALB, da dieser nur als Vermittler zwischen dem Anwender und den zu Verfügung stehenden Servern dient. Der Nachteil dieses Aufbaus ist, dass falls der Anwender sich einen Server merkt (bookmarked), dann merkt er sich einen explizieten Server, zu dem er zugewiesen wurde. Das heißt, der Benutzer merkt sich einen Server expliziet, welcher bei seinem nächsten Besuch entweder völlig überlastet oder gar nicht mehr online sein kann. Nun wird er vom Server zum ALB verwiesen und dieser weist dem Anwender einen neuen Server zu. Diese Änderung, welche sichtbar in der URL-Zeile des Browsers nachzuverfolgen ist, könnte den Anwender verwirren.

Im Verlauf der Benutzung kann der Anwender in drei Situationen beim ALB vorbei kommen. Erstens, falls er das Programm noch nie aufgerufen hat. Oder wenn der Anwender sich wieder abmeldet und darum die Benutzerdatenbank benötigt. Falls der aktuelle Server nicht mehr erreichbar ist und ein neuer Server durch den ALB herausgefunden werden muss.

3.3 Datenbank

Dieses Kapitel beschreibt den **Database Master/Peer (DBM)** und **Slave/-Peer (DBS)**. Der persistene Speicher der Cloud kann über verschiedenste Projekte realisiert werden. Damit man verschiedene Speichertypen verwenden kann benötigt man eine Abstraktionsebene, welche über den DBM, bzw. dem DBS realisiert ist. Die aktuelle verfügbaren Datenbanken sind als persistenter Speicher nutzbar:

- HBase
- Hypertable
- MySQL
- Cassandra
- Voldemort
- MongoDB
- MemcacheDB

Der DBM verwaltet alle Datenbank-Instanzen (die DBS). Für jeden Server (Kapitel 3.4) gibt es eine eigene Datenbank-Instanz, den DBS.

Die Funktionalität der Datenbankschnittstelle ist auf eine Grundfunktionalität beschränkt. Zu diesen zählt, das Hinzufügen eines neuen Elements in eine Tabelle. Falls diese definierte Tabelle noch nicht exestiert, wird diese Tabelle zuerst angelegt. Außerdem kann eine Element über eine eindeutige ID abgefragt werden. Über diese ID kann ein Element auch gelöscht werden. Zur Suche wird eine Sprache ähnlich zu SQL verwendet.

3.4 AppServer (AS)

Die eigentlich Anwendung wird auf einem AppServer (AS) ausgeführt. Diese Server wird typischerweise öfters in der Cloud zu Verfügung gestellt. Der AppLoadBalancer (Kapitel 3.2) weist den Server Benutzer zu, welche dieser dann abarbeitet. Diese Komponente basiert auf der „dev_appserver.py“ von Google. Dies ist ein kleiner Server für Googles-AppEngine-Anwendungen, welcher normalerweise auf einem Entwicklungsrechner läuft. Bei Google wird als „Datenbank“ eine Datei verwendet. Im Gegensatz hierzu, nutzt AppScale die generische Datenbankschnittstelle (Kapitel 3.3) um die Daten der Anwendungen zu speichern.

Die Speicherung lokaler Daten, wie zum Beispiel ein Authenifizierungs-Schlüssel für eine Anwendungs-Sitzung, wird lokal bei dem genutzten AppServer festgehalten.

4 Aufbau einer Cloud

Wenn man eine AppScale-Cloud aufbauen will, muss man sich zu erste im Klaren sein, ob man ein Produktiv- oder nur ein Test-System aufbauen möchte. Für ein Testsystem benötigt man nur eine AppScale-Instanz, das heißt alle vorgestellten Komponenten laufen auf einer einzigen Maschine. Eine Anleitung zu dieser Installation befindet sich auf der Projektseite [1]. Bei einem Produktiv-System benötigt man mindestens 3 AppScale-Instanzen. Eine AppScale-Instanz fungiert als Controller, das heißt hier sind AppController und AppLoadBalancer gehostet. Die anderen beiden Instanzen bearbeiten die Anfragen der Anwender. Das heißt die AppServer sind auf diesen Instanzen aktiviert.

4.1 Allgemeiner Aufbau

In einem fiktiven Aufbau, welcher aus einem Controller (head node) und einem Server besteht, können die in Abbildung 2 dargestellten Szenarien visualisiert werden. Zum einen beschreibt es das Hinzufügen einer neuen Anwendung durch einen Entwickler. Außerdem kann man die Kommunikation des Anwenders mit der Anwendung erkennen.

Entwickler Mit Hilfe der AppScale-Tools [2] kann ein Entwickler (Developer) die komplette AppScale-Cloud definieren, starteten und Anwendungen hinzufügen. Die komplette Cloud wird duch den AppController verwaltet, sodass der Entwickler hier eine neuen Anwendung hinzufügen kann. Von diesem Server (Head node) aus wird dann die Anwendung auf die weiteren Server der Cloud (nur App node in diesem Fall) kopiert, wo der Anwender (End User) diese Anwendung benutzten kann.

[1] `http://code.google.com/p/appscale/wiki/Single_Node_AppScale_Deployment`
[2] `http://code.google.com/p/appscale/wiki/AppScale_Tools_Usage`

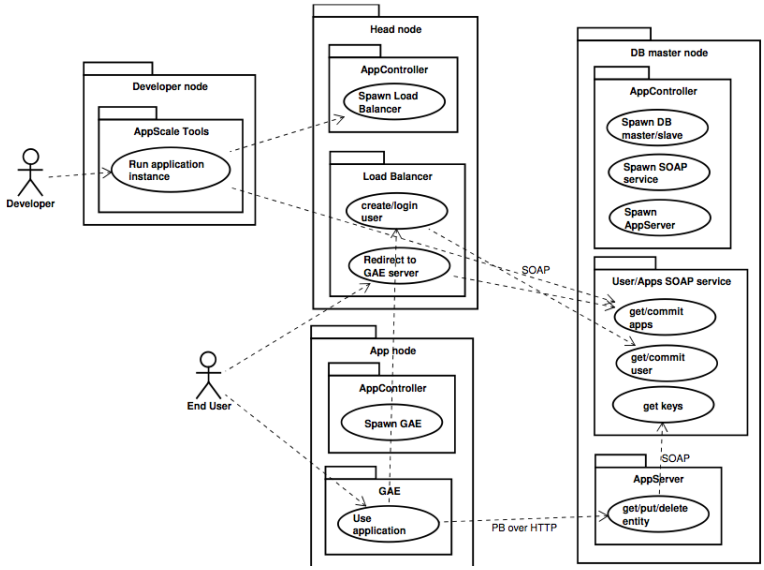

Abbildung 2. Use-Case einer AppScale-Installation

Anwender Wenn der Anwender eine Anwendung über HTTP aufrufen will, dann benötigt er einen Server (App node). Diesen erfragt er über den Controller (head node), welcher ihn zur Server (App node) umleitet. Dies ist an den beiden gestrichelten Linien, welche beim Anwender beginnen, zu sehen. Es besteht aber immer nur eine Kommunikation zu **einem** Server. Dies ist eine direkte Kommunikation, das heißt der Controller wird nach der Zuweisung des Anwenders nicht weiter beschäftigt und leitet keine Daten weiter. (Siehe Kapitel 3.2)

4.2 Aufbau in der Hochschule Mannheim

Der Aufbau an der Hochschule Mannheim besteht aus einem alten Desktop-Rechner. Dieser besteht aus einer 2.4 GHz Prozesser mit zwei Kernen und verfügt über 3GB Arbeitsspeicher. Die Festplatte hat eine Kapazität von 250GB.

Diese Ressourcen bieten Platz für 3 AppScale-Instanzen, welche in der Version 1.3 installiert wurden. Der hauptsächlich beschränkende Faktor war der Arbeitsspeicher, da die von der Projektseite angegebenen 512MB sehr knapp bemessen sind.

Da die komplette Cloud auf einer Hardware realisiert wurde, musste diese zuerst mit einer Virtualisierung ausgestattet werden. Diese Installation nutzt Xen in der Version 3.4 auf einem Debian-Lenny Betriebssystem. Jeder der virtuellen Maschine wurde 768MB Arbeitsspeicher bereit gestellt.

Außerdem musste auf der Betriebssystem des Servers (dom0) ein Webbrowser installiert werden. Dies ist von Nöten, da der Rechner nur eine im Netzwerk erreichbare IP-Addresse besitzt. Da aber jede AppScale-Instanz eine eigene IP-Addresse benötigt und der AppLoadBalancer den Anwender (Browser) direkt zu dieser IP-Addresse weiterleitetet, muss der Anwender alle AppScale-Instanzen direkt ansprechen können. Der Aufbau der virtuellen Maschinienen innerhalb des Rechners ist in Abbildung 3 zu sehen.

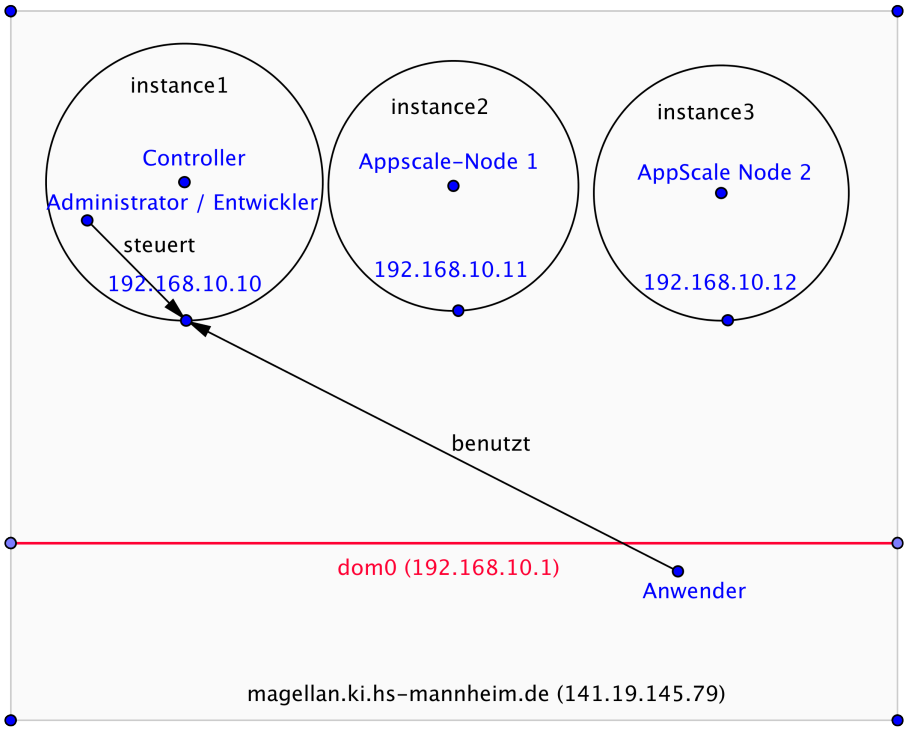

Abbildung 3. Visualisierung der Installation in Mannheim

5 AppScale-Tools

Für den Entwickler oder den Administrator der AppScale-Cloud werden die AppScale-Tools bereitgestellt. Dies ist eine Sammlung von sieben ruby-Skripten mit denen man die komplette AppScale-Cloud administrieren kann. Eine Auflistung mit einer kurzen Beschreibung ist in Tabelle 1 zusammenfasst. Die Authentifizierung zu den Server basiert auf SSH-Schlüsseln, sodass nur das erste Mal beim Einrichten der Server die Passwörter für die Root-Benutzer eingegeben werden müssen.

Tabelle 1. Alle Skripte des AppScale-Tools

#	**Befehl**	**Beschreibung**
1	appscale-add-keypair	Erstellt SSH-Schlüssel für jede Server Instanz
2	appscale-describe-instances	Beschreibt Status aller Server Instanzen
3	appscale-remove-app	Löscht ein Programm von allen Servern
4	appscale-reset-pwd	Administrator-Passwort setzen
5	appscale-run-instances	Alle Server Instanzen starten
6	appscale-terminate-instances	Alle Server Instanzen stoppen
7	appscale-upload-app	Progamm auf allen Servern installieren

Die AppScale-Cloud, welche unter Abbildung 3 zu sehen ist, wird über folgende Datei beschrieben:

Listing 1.1. ips.yaml

```
---
:controller: 192.168.10.10
:servers:
- 192.168.10.11
- 192.168.10.12
```

In Listing 1.1 sieht man wieder, dass der Server mit der IP-Addresse 192.168.10.10 die komplette Cloud verwaltet und es zwei Server gibt, welche die Anfragen der Anwender abarbeiten.

Achtung: Um möglichen Problemen bei der Ausführung der AppScale-Tools aus dem Weg zu gehen, wird empfohlen diese auf dem Controller (Head node) der Cloud zu installieren und von dort die Administration vor zu nehmen.

5.1 Starten der Cloud

Um die Konfiguration mit der Beispielanwendung „guestbook.tar.gz“ zu starten sind nur zwei Befehle nötig. Die Anwendung ist schon in dem AppScale-Tool

Packet enthalten und ist von dort in das aktuelle Arbeitsverzeichniss herauszukopieren. Nachdem man eine Konfigurationsdatei wie in Listing 1.1 erstellt hat, kann man die Cloud sammt der Beispielanwendung mit den beiden folgenden Aufrufen starten:

Listing 1.2. Starten der Guestbook-Anwendung

```
1  dev@image0:~$ appscale-add-keypair --ips ips.yaml
2  ....
3  dev@image0:~$ appscale-run-instances --file guestbook.tar
       .gz -table voldemort --ips ips.yaml
4  Head node successfully created at 192.168.10.10. It is
       now starting up voldemort via the command line
       arguments given.
5  Generating certificate and private key
6  Starting server at 192.168.10.10
7  Please wait for the controller to finish pre-processing
       tasks.
8
9  This AppScale instance is linked to an e-mail address
       giving it administrator privileges.
10 Enter your desired administrator e-mail address: f@f.d
11 Please repeat your e-mail address to verify: f@f.d
12
13 The new administrator password must be at least six
       characters long and can include non-alphanumeric
       characters.
14 Enter your new password:
15 Enter again to verify:
16 Please wait for AppScale to prepare your machines for use
       .
17 Copying over needed files and starting the AppController
       on the other VMs
18 Starting up Load Balancer
19 Done starting up AppScale, now in heartbeat mode
20
21 Your user account has been created successfully.
22 Uploading guestbook...
23 We have reserved the name guestbook for your application.
24 guestbook was uploaded successfully.
25 Please wait for your app to start up.
26
27 Your app can be reached at the following URL: http
       ://192.168.10.10/apps/guestbook
28 The status of your AppScale instance is at the following
       URL: http://192.168.10.10/status
```

Während des ersten Befehls aus Listing 1.2 muss man sich bei all den in der AppScale-Cloud verwendeten Server anmelden. Die E-Mail Adresse, welche beim Starten der Cloud (zweiter Befehl) eingegeben wird, dient zur Authentifizierung und ist gleichzeitig auch als Benutzer in die Benutzerverwaltung eingetragen worden. Das Programm überprüft sowohl das „@“, als auch den Punkt, sodass die oben eingegeben Adresse schon das Minimum darstellt.

6 Alternativen

Es gibt noch neben der App-Engine von Google ein weiteres Projekt, um GAE-Anwendungen auszuführen. Diese Projekt nennt sich AppDrop [3], welches speziell für Amazon EC2 erstellt wurde. Das Projekt wurde als proof-of-concept innerhalb von vier Tagen entwickelt. Genau wie die Entwicklungsversion von Google's App-Engine werden alle Daten in einer Datei gespeichert. Außerdem kann eine Anwendung immer nur genau auf einem Server ausführt werden. Bei dieser Konfiguration geht die gewonnene Flexebilität und Datenredundanz von Google's AppEngine oder AppScale verloren.

7 Schlusswort

Das AppScale Projekt bietet eine interessante Laufzeitumgebung für GAE-Anwendungen, welches flexsiebel zusammengestellt werden kann. Hieraus ergibt sich auch gleich das erste Problem, dass man Rechner benötigt, auf welchen die Server oder die Virtualisierung ausführt wird. Wenn man Rechner in Betrieb genommen hat, benötigt man auch Administratoren, welche sich darum kümmern. Da das Projekt in einem frühen Status ist und es häufig zu neuen Versionen kommt, benötigen die Administratoren viel Zeit um immer die neuste Version in das System einzuspielen. Bei AppScale handelt es sich um ein komplexes Softwaresystem, welches auf mehrere Rechner verteilt ist, was das Administrieren nicht erleichtert, vorallem nicht das Finden und Beheben von Fehlern.

Sobald die Server zu einer AppScale-Cloud verbunden wurde, kann eine Anwendung, die für Google's AppEngine geschrieben wurde, ohne nennenswerten Aufwand in diese Cloud installiert werden. Dadurch, dass die Cloud aus mehreren kleineren Servern besteht, lässt sich das komplette System immer wieder Stück für Stück erweitern. Da man das ganze System selbst aufgesetzt hat, hat man die komplette Kontrolle über das System und man kann den Zugriff auf seine Daten kontrollieren. Außerdem ist dies eine Möglichkeit eine GAE-Anwendung ohne Internetverbindung für einen größeren Benutzerkreis mit einer eigenen Benutzerverwaltung zu Verfügung zu stellen. Da man eine aktive Entwicklung bei diesem Projekt sieht kann man häufig von neuen Versionen erfreut werden.

[3] `http://jchris.mfdz.com`

Im Unterschied zu Google's AppEngine gibt es bei AppScale keine zeitliche Begrenzung von Berechnungen auf 30 Sekunden. Dies hat den Vorteil, dass man auch längere Berechnungen ausführen kann. Aber natürlich auch den Nachteil, dass bei einer Endlosschleife ein Server für immer beschäftigt ist und durch den Administrator reaktiviert werden muss.

Dieses Projekt kann die Abhängigkeit zu PaaS Anbietern aufweichen, in diesem konkreten Fall die Abhängigkeit zu Google. Momentan ist das System in der aktiven Entwicklung und solang diese Quelle nicht versiegt, kann sich aus diesem Projekt etwas entwickeln. Denn das Problem ist, dass es sich bei dem Gegenspieler um Google handelt, welches einen rießigen Topf an Ressourcen hat und nicht still stehen bleiben wird.

Literatur

1. Projektbeschreibung bei UCSB.
   `http://appscale.cs.ucsb.edu`
2. Server in den Wolken - AppScale.
   `http://serverwolken.de/appscale-open-source-appengine-827/`
3. Wikipedia - Artikel.
   `http://en.wikipedia.org/w/index.php?title=AppScale&oldid=331803388`
4. AppScale bei Google-Code.
   `http://code.google.com/p/appscale/`
5. AppScale - Community bei Google-Code.
   `http://groups.google.com/group/Appscale_Community`
6. Xen Hypervisor. `http://www.xen.org`
7. Kernel Based Virtual Machine.
   `http://www.linux-kvm.org`
8. Amazon EC2.
   `http://aws.amazon.com/ec2/`
9. Eucalyptus.
   `http://www.eucalyptus.com/`
10. Ruby on Rails.
    `http://rubyonrails.org/`
11. nginx - Webserver.
    `http://nginx.org/`