Cloud Computing - Google App Engine

Peter Sutter

Fakultät für Informatik
Hochschule Mannheim
Paul-Wittsack-Straße 10
68163 Mannheim
peter.sutter@stud.hs-mannheim.de

Zusammenfassung Google stellt mit App Engine einen Dienst zur Verfügung der sich nicht wie die bisherigen Produkte an die breite Masse der Internetnutzer richtet, wie zum Beispiel mit „Google Docs“, sondern eher an Webentwickler. Diese Arbeit stellt Google App Engine vor, sowie die zur Verfügung stehenden Tools und Services, welche die Entwicklung von Anwendungen unterstützen bzw. erleichtern sollen, sowie die Regeln die einer Anwendung in Google's Cloud vorgeschrieben werden.

Steht man vor einem neuen Software Entwicklungsprojekt im Web-Bereich, muss man sich unter anderem mit der Fragestellung auseinandersetzen, wo die spätere Applikation laufen soll. Baut man entweder eine eigene Infrastruktur auf, das heißt das fertige Produkt läuft auf einem eigenen Server für dessen Hardware, Betrieb, Wartung und Skalierung Kosten bzw. Aufwände aufgebracht werden müssen oder mietet man sich einen Webspace, für den man monatlich einen festen Betrag zahlt oder aber setzt man auf PaaS Anwendungen wie zum Beispiel Google App Engine, bei der man seine Anwendung Hosten kann? Diese Ausarbeitung erläutert kurz die Vor- und Nachteile der einzelnen Alternativen, anschließend wird näher auf Google App Engine eingegangen.

1 Vergleich mit herkömmlichen Alternativen

Beim Aufbau einer eigenen IT-Infrastruktur muss man mehrere Kostenpunkte im Auge behalten. Zum einen wären dies hohe Kapitalinvestitionen und sprungfixe Kosten, wie die Beschaffung der Serverhardware, falls noch nicht vorhanden oder der Aufrüstung bereits existierender Komponenten, um den Lastanforderungen der neuen Applikation gerecht zu werden. Zum anderen führt dies

> ... zu redundanten Ausgaben und Überkapazitäten, sowohl was die Technologie selbst als auch die für ihren Betrieb benötigten Arbeitskräfte angeht.[2][1]

[1] S. 25

Positiv dem gegenüber steht die Erschaffung neuer Arbeitsplätze im eigenen Unternehmen, was in der heutigen Zeit nicht hoch genug angerechnet werden kann.[2] Jedoch bleibt den Unternehmen meist keine andere Wahl als dabei Millionen von Angestellten zu entlassen um Kosten zu sparen.

Der Datenschutz spielt für die gespeicherten Daten einer Anwendung, wie z. B. der Kundendaten eine große Rolle. Sind diese entweder auf den eigenen Speichersystemen gelagert, muss man selbst für die hohen Sicherheitsvorkehrungen aufkommen oder man überlässt die Speicherung der Daten dem externen Dienstleister und muss dafür hohes Vertrauen in diesen setzen.

Ein weiterer Vorteil einer eigenen IT-Infrastruktur ist die Unabhängigkeit gegenüber Webhoster, denn hat man sich für einen Webhoster als Plattform für das eigene Produkt entschieden, muss man mit dem Risiko rechnen, dass der Hoster die Preise erhöht oder seinen Dienst einschränkt oder einstellt. Ein Umzug zu einem anderen Hoster mit ähnlicher Angebotspalette kostet dann erneuten Konfigurationsaufwand, eine Änderung an der Applikation ist aber nicht nötig. Aufgrund von fixen monatlichen Beträgen ist zudem eine gute Überschaubarkeit und Kalkulierbarkeit der Kosten gegeben.

Fällt die Wahl des Providers auf Google App Engine, so profitiert man im Gegensatz zu den zuvor genannten Alternativen eine sehr gute Skalierung, ebenso wie hohe Verfügbarkeit und Performanz, dank Googles gewaltiger Infrastruktur.[3][3] Diese guten Leistungen erhält man sogar für relativ geringe Kosten und man zahlt nur für die Leistung, die man in Anspruch genommen hat.

> The cost of purchasing resources from Google's cloud of servers is likely far less than purchasing/renting/maintaining the same amount of resources on your own.[3][4]

Näheres zu diesem Thema findet sich in Kapitel 2.6.

Google App Engine bietet außerdem nützliche Services an die der Entwickler in seine Anwendung integrieren kann, wie zum Beispiel der Images API zur Bildermanipulation oder die Login-Authentifizierung über Google Accounts. Allerdings muss man für diese Unterstützung einen Kompromiss eingehen, denn man gibt sich dadurch in Abhängigkeit von Google App Engine. Möchte man die eigene Anwendung aus bestimmten Gründen nicht mehr in Googles Cloud laufen lassen, muss man entweder diese Google App Engine spezifischen Dienste selbst implementieren oder durch andere Bibliotheken ersetzen. Zudem müsste man einen eigenen Server bzw. eine private Cloud mit Googles Open Source Framework AppScale[5] aufsetzen. Damit lassen sich die Google App Engine Web Anwendungen ausführen. Man muss aber mit einem hohen Konfigurationsaufwand, spärlicher Dokumentation und einer kleinen, mehr oder weniger aktiven

[2] Vgl. Pressetext: http://pressetext.de/news/091022041/krise-kostete-3-2-millionen-jobs-in-einem-jahr/
[3] S. 8
[4] S. XI

Community rechnen.[5] Jedoch bemüht sich Google bei diesen Schnittstellen auf offene Standards zu setzen, damit der Einstieg für Entwickler erleichtert wird. Als Beispiel kann hierfür die Java Mail-, Memcache- und URL Fetch API genannt werden.

Web Anwendungen laufen bei Google App Engine aus Sicherheitsgründen in einer Sandbox, deren Regeln sie befolgen müssen. Diese Einschränkungen haben die Folge, dass nicht alle möglichen Anwendungstypen in Google's Cloud gehostet werden können, ohne Teile davon auszulagern.

2 Google App Engine

2.1 Was ist Google App Engine

Mit App Engine hat Google im April 2008 eine „Platform as a Service" der Öffentlichkeit zur Verfügung gestellt, welche sich aktuell noch im Beta-Stadium befindet. Auf diese Cloud Computing Plattform können Entwickler ihre Web Anwendungen deployen und ausführen, welche auf Google's Infrastruktur gehostet werden. Google kümmert sich automatisch um die Lastverteilung und die Skalierung der Anwendung. Dies erfolgt völlig transparent für den Entwickler bzw. den Benutzer. Ein direkter Einfluss darauf, z. B. auf wievielen Servern oder in welchem Datenzentrum die Applikation ausgeführt werden soll, ist somit nicht möglich. Zielgruppe für App Engine sind Anwendungen mit kurzen Antwortzeiten, denn erfolgt die Antwort auf einen Request nicht innerhalb von 30 Sekunden wird eine *DeadlineExceeded* Exception geworfen und anschließend der Request Handler beendet.

2.2 Laufzeitumgebungen

Unterstützt wird, seit der Veröffentlichung, die Programmiersprache Python. Die Python Laufzeitumgebung von App Engine setzt auf Python in der Version 2.5.2. Die derzeit aktuelle Python Version 3.x wird noch nicht unterstützt. Genau ein Jahr später wurde auch der Java Sprachsupport den Entwicklern zur Verfügung gestellt. Google App Engine nutzt die Version 6 der *Java Virtual Machine* (JVM). In dieser laufen Anwendungen, die in Java 5 oder Java 6 kompiliert wurden, nicht aber in Java v1.4. Darüber hinaus werden auch solche Sprachen unterstützt, die auf der JVM basieren. Darunter fallen zum Beispiel die Programmiersprachen Scala, Groovy, JRuby (Ruby) oder Quercus (PHP).

2.3 Java EE Unterstützung

Java Anwendungen werden wie gewöhnlich als *Web Application Archives* (WAR) verpackt und auf Google's Application Server deployed. Für Java EE Entwick-

[5] Vgl AppScale Community: http://groups.google.com/group/appscale_community/

ler stellt sich nun die Frage, ob sie alle Funktionen der *Java Enterprise Edition* nutzen können.[6] Leider wird nur eine Teilmenge aller Java EE Funktionen unterstützt. So lassen sich keine entfernten Methodenaufrufe mittels RMI ausführen, jedoch könnte man als Alternative den *URL Fetch* Service von App Enginge verwenden, um z. B. mit RESTful Webservices zu interagieren. Ebenso steht das *Java Naming and Directory Interface* (JNDI) nicht zur Verfügung, um Referenzen auf entfernte Objekte zu erlangen. Da *Enterprise Java Beans* (EJB) ebenfalls über das JNDI nachgeschlagen werden, ist eine Nutzung dieser ebenso ausgeschlossen. Datenbank Verbindungen über JDBC sind nicht möglich, dafür lassen sich Daten mit den High-Level Schnittstellen JPA oder JDO modellieren und in Google's BigTable Datenbank persistieren. App Engine nutzt einen von Google entwickelten Adapter um das Open Source Framework *Datanucleus*[6], als Implementierung der beiden Persistenz-Schnittstellen, anzubinden. Als Schnittstelle zum Webbrowser können die JSP-, JSF- und Servlet-Technologien eingesetzt werden, die in einem modifizierten Jetty[7] Servlet-Container zum Einsatz kommen.

> The key features that Google would have chosen jetty for were size and flexibility.[7]

Da mehrere tausend Instanzen von Jetty in Google's Cloud laufen, schlägt sich jedes eingesparte MB pro Server positiv auf die zur Verfügung stehenden Ressourcen für Anwendungen aus. Neben Jetty's geringem Speicherbedarf spricht zusätzlich noch dessen Erweiterbarkeit, somit ist eine einfache Anpassung an Google's Bedürfnisse gegeben.

2.4 Restriktionen der Sandbox

Damit sich die Anwendungen auf einem Server nicht gegenseitig stören oder beeinflussen können, werden diese in einer Sandbox ausgeführt. Diese Sandbox schreibt einige Regeln vor, an die sich die Anwendungen halten müssen. Diese Regeln werden im Folgenden erläutert.

Viele dieser Restriktionen kennt man bereits aus der Java EE Welt. Zum einen dürfen im Anwendungsserver keine Threads gestartet werden. In der EJB Spezifikation wird folgender Grund genannt, der auch auf Google App Engine übertragen werden kann.

> Allowing the enterprise bean to manage threads would decrease the container's ability to properly manage the runtime environment.[8]

Als Alternative zu Threads bietet App Engine den Task Queue Service an, um asynchron Arbeiten zu verrichten. Zum anderen darf nicht auf das lokale Datei-

[6] Vgl. Datanucleus App Engine: http://www.datanucleus.org/products/accessplatform_1_1/appengine/support.html
[7] Vgl. Jetty Homepage: http://www.mortbay.org/jetty/

system geschrieben werden. Alternativ kann man die Daten persistent im Datastore oder flüchtig im Memcache ablegen. Dateien, die mit der Anwendung deployed wurden dürfen jedoch gelesen werden. Weiterhin dürfen keine Sockets aus der App Engine Anwendung geöffnet werden, zum Beispiel um direkte Netzwerkverbindungen aufzubauen. Google bietet hierfür den URL Fetch Service an, um mit anderen Webseiten zu kommunizieren. Wie schon zuvor erwähnt ist die Antwort auf einen HTTP-Request auf 30 Sekunden beschränkt. Zudem sind Request und Response mit bis zu 10MB erlaubt.

Da nicht alle Klassen aus der JRE verwendet werden können, hat App Engine eine Whitelist[8] aller erlaubten Klassen veröffentlicht. Es fehlen z. B. die meisten Klassen aus dem awt Package zum Erstellen von User Interfaces oder dem rmi Package um entfernte Methodenaufrufe auszuführen. Insgesamt sind nur ca. 40% aller Klassen der JRE verfügbar.

Wie auch Java unterliegt Python den selben Einschränkungen. So wurden bei Python manche Module aus der Standard Library deaktiviert, die gegen die Sandbox-Regeln verstoßen. Das Verwenden von Third Party Libraries für Python ist gestattet, sofern diese kein C-Quellcode enthalten und komplett in Python geschrieben sind.

2.5 Tools

App Engine bietet verschiedene Tools an um den Entwicklern die Arbeit zu erleichtern. Darunter zählt das Eclipse Plugin, der Google App Engine Launcher, der Development Server und die Adminkonsole, die nachfolgend vorgestellt werden.

Für Java Anwendungen wurde ein Eclipse Plugin geschrieben. Damit lassen sich über einen Wizard Web Application Projekte erstellen. Es wird eine Beispielanwendung erstellt, ebenso wie alle benötigten Konfigurationsdateien, wie zum Beispiel der *appengine-web.xml*. Außerdem lässt sich mit dem Plugin die Anwendung über ein GUI auf App Engine deployen.

Für Python gibt es den sogenannten *Google App Engine Launcher*, zu sehen in Abbildung 1. Mit diesem Tool lässt sich der mitgelieferte *Development Server* starten und stoppen, die Anwendung auf App Engine deployen, lokale Server Logs und die *Development Console* vom Development Server einsehen.

Der *Development Server* ist eine Nachbildung des Anwendungsservers, der in App Engine zum Einsatz kommt. Dieser Server ist für Entwicklungszwecke geeignet. Man spart sich die Zeit die benötigt wird, um die Software auf App Engine hochzuladen. Zudem arbeitet man mit lokalen Testdaten und läuft somit nicht in Gefahr die Produktivdaten durch fehlerhafte Software zu gefährden. Da dieser Entwicklungsserver kein 1:1 Clon ist kann es sein, dass manche Funktionen nicht auf dem Entwicklungsserver, aber in App Engine laufen oder umgekehrt. In der Development Console des Entwicklungsservers lässt sich unter anderem der

[8] http://code.google.com/intl/de-DE/appengine/docs/java/jrewhitelist.html

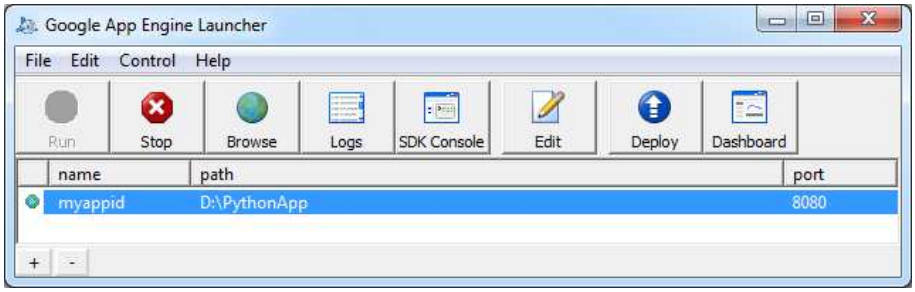

Abbildung 1. Google App Engine Launcher für Python

lokale Datastore oder der Inhalt des Memcaches anzeigen. Erreichbar ist diese Web-Oberfläche unter `http://localhost:8080/_ah/admin/`.

Ein Tool zum Verwalten der eigenen Anwendungen ist die Adminkonsole[9]. Mit dieser lassen sich neue Anwendungen erstellen, d.h. es wird ein eindeutiger Name reserviert, unter dem die Anwendung später erreichbar sein wird. Die URL lautet dann `http://<myappid>.appspot.com/`. Es lassen sich maximal 10 Anwendungen erstellen. Seit Oktober 2009[10] kann man erstellte Anwendungen in der Adminkonsole auch wieder löschen und man erhält eine Gutschrift zum Erstellen einer neuen Applikation. Der vergebene Applikationsname kann jedoch nicht mehr wieder verwendet werden. In der Adminkonsole lassen sich außerdem die Log-Einträge, die verbrauchten Ressourcen, sowie die noch verfügbaren Ressourcen einsehen. Die verbrauchten Ressourcen werden einmal am Tag zurückgesetzt. Google App Engine ist zu einem gewissen Grad kostenlos. Falls die eigene Anwendung im Laufe der Zeit populärer wird und die kostenlos zur Verfügung stehenden Ressourcen nicht mehr ausreichen, kann man zusätliche Ressourcen wie z. B. CPU-Zeit, Bandbreite, Speicherplatz, etc. zukaufen. Mehr dazu in Kapitel 2.6. Wie auch in der Development Console kann man in der Adminkonsole die gespeicherten Daten im Datastore oder den Status der Task Queue anzeigen lassen. Weiterhin lassen sich die Versionen der Anwendung verwalten. Man kann dort eine Version als *Default* markieren. Dies bedeutet, dass nun alle Anfragen an die URL `http://<myappid>.appspot.com/` an diese Default-Version geleitet werden. In der Adminkonsole kann man auch weitere Entwickler für eine Applikation hinzufügen. Diese haben dann z. B. das Recht neue Versionen der Anwendung hochzuladen, oder diese vollständig zu löschen.

[9] https://appengine.google.com/

[10] http://googleappengine.blogspot.com/2009/10/app-engine-sdk-126-released-with.html

2.6 Ressourcen und Preise

Google App Engine schreibt in dessen offiziellen Blog folgendes über die kostenlose Nutzung von App Engine:

> Our target level of free support has been 5 million page views per month for a reasonably efficient web application.[11]

In Tabelle 1 wird das täglich neu zur Verfügung stehende freie Kontingent aufgelistet, ebenso wie die Kosten der jeweiligen Ressource pro Einheit bei Überschreitung des Limits.

Tabelle 1. Kostenlos zur Verfügung stehende Ressourcen

Resource	Unit Cost	Free Quota
CPU Time	$0.10/CPU hour	6.50 CPU hours
Outgoing Bandwidth	$0.12/GByte	1.00 GBytes
Incoming Bandwidth	$0.10/GByte	1.00 GBytes
Stored Data	$0.005/GByte-day	1.00 GBytes
Recipients Emailed	$0.0001/Email	2000

Dieses kostenlos und täglich neu zur Verfügung stehende Budget soll also, laut Google's Einschätzung, für etwa 166 tausend Seitenaufrufe pro Tag ausreichen und ist mit den aktuellen Preisen von App Engine insgesamt etwa $1 wert.

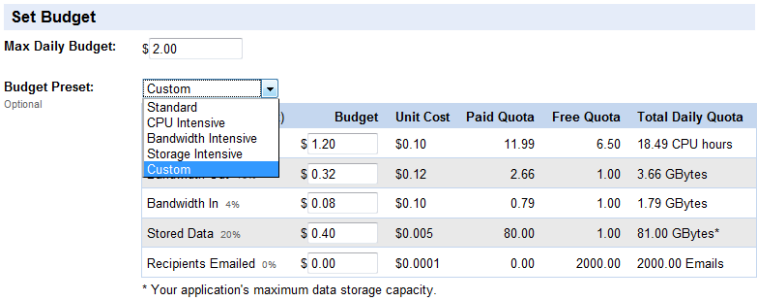

Abbildung 2. Adminkonsole - Budget festlegen

Standardmäßig ist die Bezahlung deaktiviert und kann wenn gewünscht in der Adminkonsole unter *Billing Settings* aktiviert werden. Festgelegt wird ein Budget, für das man maximal bereit ist pro Tag zu bezahlen. Dieses Budget kann,

[11] http://googleappengine.blogspot.com/2009/06/changing-quotas-to-keep-most-apps.html

wie in Abbildung 2 zu sehen, beliebig auf die Ressourcen verteilt werden. Man zahlt erst für die Ressource, wenn das freie Kontingent aufgebraucht ist.

2.7 Konfiguration einer Anwendung

In Kapitel 2.5 wurde beschrieben, dass man in der Adminkonsole eine neue Anwendung erstellen bzw. dessen eindeutige Identifikation reservieren kann. Wurde dieser Schritt ausgeführt kann man damit beginnen die Anwendung auf App Engine vorzubereiten. Hierfür ist das Erstellen ein oder mehrerer Konfigurationsdateien vonnöten. Im Folgenden wird dies für Python und Java aufgeführt.

Python Erstellt man eine Anwendung in Python für App Engine, muss man eine Konfigurationsdatei mit dem Namen *app.yaml* im Stammverzeichnis der Anwendung anlegen.

Listing 1.1. app.yaml - Python App Engine Konfigurationsdatei

```
application: application-id
version: 1
runtime: python
api_version: 1
handlers:
- url: /.*
  script:myApp.py
```

In Listing 1.1 wird der Inhalt dieser Konfigurationsdatei exemplarisch aufgezeigt. Unter *application* wird der Name der Anwendung eingetragen, der in der Adminkonsole erstellt wurde. Unter *version* trägt man eine beliebige alphanumerische Version ein. Der Parameter *runtime* gibt die Laufzeitumgebung Python an. Zur Zeit ist hier nur die Angabe von Python möglich, denn für Java Anwendungen wird eine andere Konfigurationsdatei benötigt. Der vierte Parameter *api_version* gibt die verwendete API Version von Google App Engine an, die sich aktuell in der Version 1 befindet. Anschließend wird mit dem Parameter *handlers* die Abbildung von URLs auf Skripte definiert. In diesem Beispiel werden alle Anfragen an das Python-Skript *myApp.py* geleitet.

Java Erstellt man eine Java Webanwendung für App Engine, ist die Konfiguration auf zwei Dateien aufgeteilt. Die App Engine spezifischen Konfigurationen werden in einer XML-Datei namens *appengine-web.xml* im *WEB-INF* Ordner gepflegt und muss mindestens die Elemente aus Listing 1.2 enthalten.

Listing 1.2. appengine-web.xml - Java App Engine Konfigurationsdatei

```
<appengine-web-app
  xmlns="http://appengine.google.com/ns/1.0">
```

```
  <application>ccseminar09</application>
  <version>vers1</version>
</appengine-web-app>
```

Die Elemente *application* und *version* sind analog zu den Python-Parametern in der *app.yaml*. Die verwendete API Version von App Engine muss hier nicht angegeben werden, da sich das verwendete App Engine SDK im *lib* Ordner der WAR-Datei befindet, z. B. *appengine-api-1.0-sdk-1.2.6.jar*. Das Mapping von URLs auf Servlets wird, wie in Java EE Anwendungen üblich, im Deployment Deskriptor namens *web.xml* gepflegt. Nachlesen kann man dies bei Bedarf auf der Dokumentationsseite von App Engine unter [9].

2.8 Versionierung

Wie in den Konfigurationsdateien im Kaptiel 2.7 zu sehen ist, kann man einer Anwendung Versionen vergeben und diese anschließend auf Google App Engine deployen. Welche Möglichkeiten und Probleme sich hierdurch ergeben wird im Folgenden durch ein Beispiel erläutert.

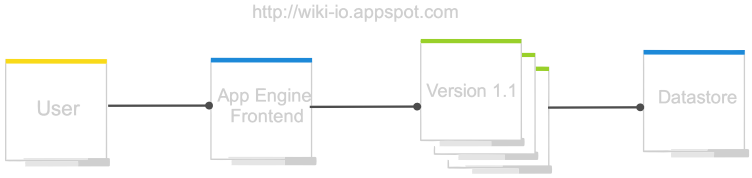

Abbildung 3. Beispielanwendung wiki-io - v1.1 [10]

Abbildung 3 zeigt eine Anwendung mit dem Namen *wiki-io* in der Version 1. Alle Anfragen des Benutzers werden an diese erste Version geleitet, da keine andere Version vorhanden ist und diese die Default-Version darstellt.

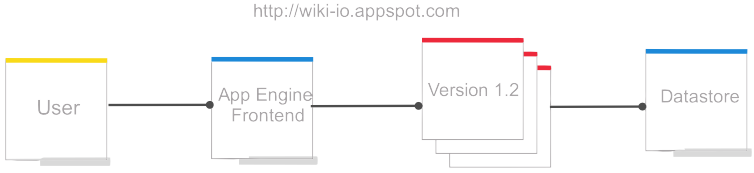

Abbildung 4. Beispielanwendung wiki-io - v1.2 [10]

Fügt man der Anwendung z. B. neue Funktionen hinzu und deployed diese mit der selben Versionsnummer, wird, wie in Abbildung 4 zu sehen, die alte Version überschrieben und mit der neuen Version ersetzt. Alle Anfragen an wiki-io werden nun an diese neue Version 1.2 geleitet. Ist dieses neue Feature noch ungenügend getestet bzw. verhält es sich auf App Engine's Server anders als auf dem lokalen Entwicklungsserver, so kann es passieren, dass Anwender Fehlerseiten angezeigt bekommen, ohne dass man vorher die Möglichkeit hatte diese zu beheben.

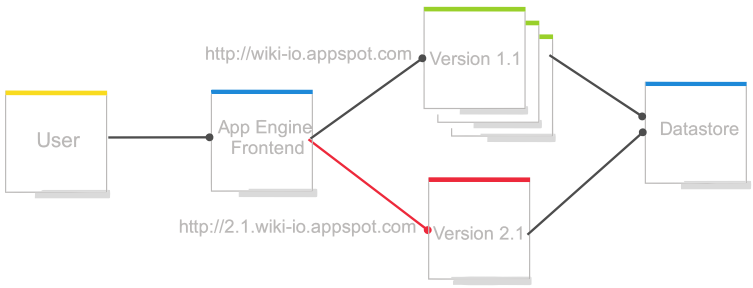

Abbildung 5. Beispielanwendung wiki-io - v2.1 [10]

Deployed man die Änderungen jedoch mit einer neuen Versionsnummer, wie in Abbildung 5, z. B. mit der Versionsnummer 2, so wird die alte Version 1 nicht überschrieben und bleibt weiterhin die Default-Version. Version 2 läuft parallel zu Version 1 und kann über eine eigene URL, welche die Versionsnummer enthält, getestet werden. Somit ist es möglich die Anwendung in der Produktivumgebung und mit Produktivdaten zu testen, während die Anwender weiterhin die stabile Version benutzen. Die Anwendung mit den Produktivdaten zu testen birgt das Risiko diese zu verlieren oder unbeabsichtigt zu verändern. Jedoch sollte eine Anwendung ausgiebig im Entwicklungsserver getestet werden, bevor sie in die Produktivumgebung deployed wird. Alternativ könnte man auch ein Backup des Datastores erstellen und dieses in eine zweite Applikation hochladen und dann dort testen. Dies lässt sich mit dem Bulkloader bewerkstelligen, welcher unter [11] beschrieben ist.

3 Google App Engine APIs

Google App Engine bietet den Entwicklern verschiedene Services an, entweder als Alternative zu Dingen, die aufgrund der beschriebenen Sandbox Restriktionen in Kapitel 2.4 nicht erlaubt sind oder um dem Entwickler nützliche Funktionen zur Hand zu geben, wie z. B. des Image-Manipulationsservices. Diese Services werden im Folgenden aufgeführt.

3.1 Datastore

App Engine setzt zur Speicherung der Daten aller Anwendungen auf BigTable. BigTable ist Googles selbst geschriebene Datenbank für schemalose, strukturierte Daten. Schemalos, da BigTable nicht zu den relationalen Datenbanken zählt und eher vergleichbar mit einer verteilten, sortierten und geschachtelten HashMap ist. Die Daten aller Anwendungen werden logisch in einer BigTable Tabelle gespeichert. Diese wird in sogenannte Tablets partitioniert und auf die Server verteilt. Ein Server ist für ca. 100 Tablets von einer Größe von etwa 100-200MB verantwortlich. BigTable skaliert horizontal, d.h. dass bei steigender Last oder Datenvolumen zusätzliche Server eingesetzt werden können. Fällt ein Server aus, übernehmen etwa 100 Server jeweils ein Tablet dieses Servers um die Last gleichmäßig zu verteilen. Wie in Abbildung 6 zu sehen ist, kommunizieren die Google Anwendungen mit BigTable über sprachspezifische Schnittstellen. Bei Java wären dies die standardisierten Schnittstellen JDO und JPA, bei Python setzt der Datastore als Schnittstelle auf BigTable.

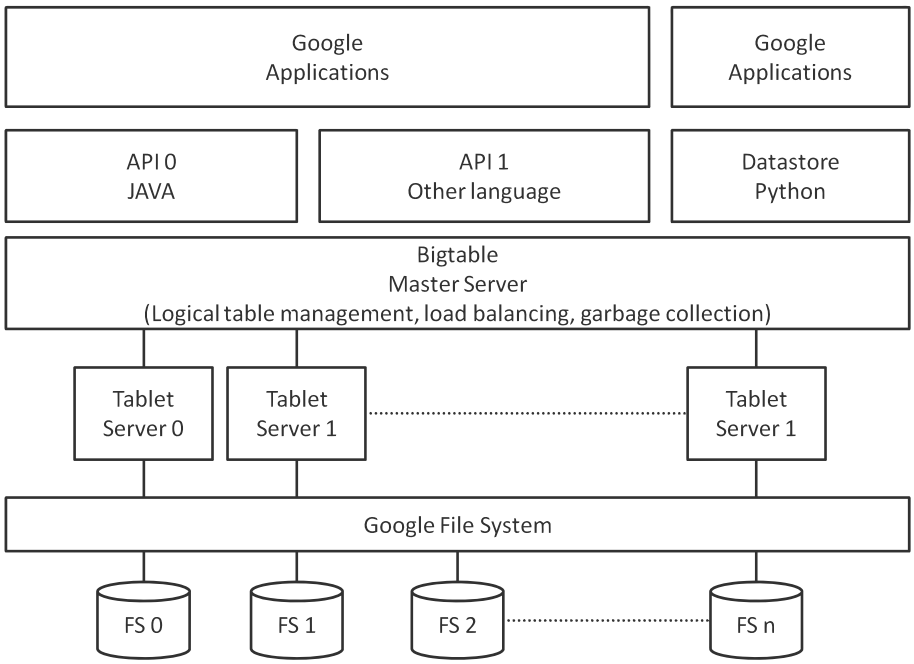

Abbildung 6. Datastore Architektur[3]

3.2 Memcache

Memcache ist im Gegensatz zum Datastore ein temporärer Speicher, der, wie auf Abbildung 7 zu sehen ist, von allen Instanzen der Anwendung geteilt wird.

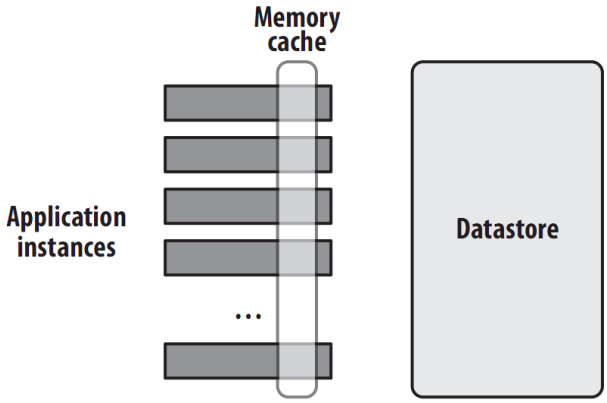

Abbildung 7. Memcache [4]

Wie der Name Memcache schon sagt befindet sich dieser im Arbeitsspeicher des Servers und hat dadurch sehr gute Zugriffszeiten, da nicht von der Festplatte gelesen werden muss. Jeder Eintrag wird im Memcache, wie bei einer HashMap, mit einem eindeutigen Schlüssel abgelegt und ist auf 1MB beschränkt. Zusätzlich kann man noch eine Verfallszeit in Sekunden angeben, wann der Eintrag aus dem Memcache entfernt werden soll. Man muss aber damit rechnen dass der Eintrag aufgrund von Ressourcenmangel schon vor Ablauf der Frist gelöscht wird. Man sollte also im Memcache nur Daten ablegen, die man auch wieder auf andere Weise besorgen kann, z. B. durch das zusätzliche Ablegen im etwas langsameren Datastore. Hierfür gibt es bereits das folgend aufgeführte Memcache Pattern[12], das sich dieses Vorgehen zu Nutze macht.

Listing 1.3. Memcache Pattern (Python)

```
def get_data():
    data = memcache.get("key")
    if data is not None:
        return data
    else:
        data = self.query_for_data()
        memcache.add("key", data, 60)
        return data
```

Das Listing 1.3 beschreibt die Methode *get_data*. Wird der Eintrag mit dem Schlüssel „key“ im Memcache gefunden, wird dieser zurückgegeben und man kann somit von dem Vorteil des schnellen Speichers profitieren. Wird der Eintrag jedoch nicht gefunden, wird nach diesem zum Beispiel im Datastore gesucht und anschließend zusätzlich im Memcache abgelegt, sodass der nächste Aufrufer dieser Funktion vom Memcache profitieren kann.

3.3 Mail

Zum Senden von E-Mails an beliebige Empfänger über das GMail Gateway lässt sich die Mail API verwenden. Der Absender der E-Mail kann jedoch aus Sicherheitsgründen nicht frei gewählt werden. Als Absender kann nur die E-Mail Adresse des zur Zeit eingeloggten Benutzers oder die E-Mail Adresse des Administrators der Anwendung genutzt werden. Das Anhängen von Dateien und mehrere Empfänger sind ebenfalls möglich. Nicht nur das Senden, sondern auch das Empfangen von E-Mails ist über diese Schnittstelle möglich. Die E-Mail wird als POST Request an `/_ah/mail/<senderadresse>` gesendet. Diese URL kann im Deployment Deskriptor auf ein Servlet gemappt werden, welches die E-Mail entgegen nimmt.

3.4 URL Fetch

Zur Kommunikation mit anderen Webseiten oder um RESTful Web Services aufzurufen lässt sich die URL Fetch Schnittstelle nutzen. Die URL muss standard Ports für HTTP (80) bzw. HTTPS (443) verwenden. Als Request Methoden werden GET, POST, PUT, DELETE und HEADER unterstützt. Request und Response sind jeweils auf 1MB beschränkt.

3.5 Google Accounts

Mit diesem Service können sich Benutzer über ihren Google Account einloggen. Man muss also selbst keine eigene Login-Logik implementieren und kann sich auf die Sicherheit dieses Dienstes verlassen, welcher auch in anderen Produkten von Google Verwendung findet, wie z. B. bei Google Kalender oder Google Mail. Ein weiterer Vorteil ist, dass sich Benutzer nicht bei der Anwendung registrieren müssen, da ein vorhandener Google Account ausreicht.

3.6 Image

Mit dem Image Service lassen sich serverseitig Bilder manipulieren. Darunter fallen die Operationen wie z. B. das Drehen oder Ausschneiden von Bildern, sowie die „I’m Feeling Lucky“-Funktion zum Verbessern der Helligkeit, Kontrast und Farben.

3.7 Task Queue

Um asynchron Arbeiten ausführen zu können, bietet App Engine als Alternative zu Threads den Task Queue Service an. Eine Anwendung kann Tasks erstellen und diese in einer Queue ablegen, die dann nach dem FIFO Prinzip abgearbeitet werden. Der „Queue mediator“ holt sich ein oder mehrere Tasks aus der Queue, je nach Einstellung, und sendet diese zur Abarbeitung an die Request Handler, wie in Abbildung 8 zu sehen. Man kann festlegen, mit welcher HTTP Request Methode dies an die Request Handler gesendet wird, ebenso wie die Parameter zum weiteren Spezifizieren der Aufgabe.

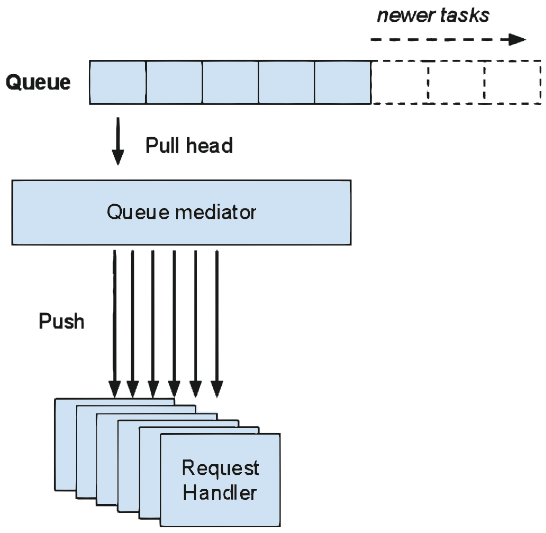

Abbildung 8. Task Queue [13]

4 Schlusswort

Mit App Engine wird dem Entwickler ein Service zur Verfügung gestellt, mit dem sich Anwendungen schnell erstellen, testen und anschließend auf Google's Infrastruktur hosten lassen. Aufgrund des freien Kontingents, welches für 5 Millionen Seitenaufrufen im Monat ausreicht, ist App Engine besonders für Applikationen interessant mit geringen Nutzerzahlen, da keine Kosten für die Anwendung entstehen. Selbst für Anwendungen mit hohen Nutzerzahlen oder schwankender Zugriffshäufigkeit ist App Engine aufgrund der fast beliebigen Skalierbarkeit und der relativ geringen Kosten ein interessanter „Platform as a Service Anbie-

ter“. Interessant auch nur dann, wenn man die Webanwendung mit den in dieser Arbeit genannten Restriktionen von App Engine auch umsetzen kann.

Literatur

1. Google AppEngine.
   http://code.google.com/appengine/
2. Nicholas Carr. *The Big Switch: Der große Wandel. Die Vernetzung der Welt von Edison bis Google*. Heidelberg: Mitp-Verlag, 1. Auflage 2009. ISBN: 978-3-8266-5508-1.
3. Eugene Ciurana. *Developing with Google App Engine*. Apress, 1. Auflage 2009. ISBN: 978-1-4302-1831-9.
4. Charles Severance. *Using Google App Engine*. O'Reilly Media, First Edition May 2009. ISBN: 978-0-596-80069-7.
5. AppScale – an open-source research framework.
   http://appscale.cs.ucsb.edu/
6. Will it play in App Engine.
   http://groups.google.com/group/google-appengine-java/web/will-it-play-in-app-engine
7. Google chose Jetty for App Engine.
   http://www.infoq.com/news/2009/08/google-chose-jetty
8. Java Specification Request - Enterprise JavaBeans 3.0
   http://jcp.org/aboutJava/communityprocess/final/jsr220/index.html
9. The Deployment Descriptor: web.xml
   http://code.google.com/intl/de-DE/appengine/docs/java/config/webxml.html
10. Ken Ashcraft - Building a Production Quality Application on Google App Engine
    http://sites.google.com/site/io/best-practices—building-a-production-quality-application-on-google-app-engine
11. Bulkloader - Downloading and Uploading All Data
    http://code.google.com/intl/de-DE/appengine/docs/python/tools/uploadingdata.html
12. The Memcache Pattern
    http://code.google.com/intl/de-DE/appengine/docs/python/memcache/usingmemcache.html
13. Offline Processing on App Engine
    http://code.google.com/intl/de-DE/events/io/2009/sessions/OfflineProcessingAppEngine.html