Google App Engine

Vortrag im Seminar CloudComputing SS 2009 von

Andreas Beyer

Institut für Informatik
Universität Heidelberg

- „We carry pagers, so you don't have to!“
  - Kurze Wege von der Idee zu Realisierung
- Ein Sandkasten in der Cloud

- Google will mehr Nutzer im Internet
  - bietet also einfachen Weg Inhalte zu generieren
- werden durch attraktives Angebot angelockt
  - sehen Googels Werbung
    - AdSense / AdWords / Analytics / DoubleClick
- Den Nutzern folgen mehr Entwickler
- profitable Entwickler überschreiten Quota
  - diese zahlen für Space/Traffic/CPU-Zeit
- . . . und schon verdient Google (mehr) Geld

- Amazon S3 war 02/2008 für zwei Stunden nicht erreichbar
- Amazon S3 (& Amazon SQS) waren 07/2008 für acht Stunden der US-Tageszeit down
- App Engine hatte 07/2008 für fünf Stunden Teilausfälle
- App Engine hatte für einige Tage das Problem hoher Latenzen Anfang Mai 2009

- Plattform zum Entwickeln und Hosten von Webanwendungen auf Google-Servern
- Biete Kapselung aller Schritte die der „normale“ Prozess fordert
- in Maßen kostenfrei
- integriert in restliches Angebot
- besteht aus Framework und sechs APIs

- selbst konfigurierbare virtuelle Maschine
  - im Unterschied zu Amazon
- FTP-Server
- Dienst an den Jobs / Berechnungen abgegeben werden können
- vollwertige relationale Datenbank
- die Neuerfindung des Rades
  - sämliche Strukturen bestehen bereits
  - werden nun Entwicklern zugänglich gemacht

- LAMP ist sogenannter Industriestandard
  - Linux aufsetzen
  - Apache installieren (& konfigurieren)
  - MySQL installieren (& konfigurieren)
  - Python/PHP/Perl installieren
  - WebApplikation in beliebiger Sprache schreiben
- „ins Internet bringen“ / testen und hosten
  - Domain erwerben & Traffic/Space bereitstellen
- überwachen (Sicherheit / Verfügbarkeit / Last)

- pro:
  - Man hat alles selbst im Griff und unterliegt keinen Beschränkungen durch Dritte (Google)
- contra:
  - Zeitintensive Einrichtung
  - Wartungsaufwand
  - Portierung bei Überlast
    - Google skaliert automatisch
  - von Anfang an Kosten

- BigTable
- Datastore
- Memcache
- APIs
- Applikation in
  - „Django & Python“
  - oder „Java (& X)“

- Mechanismen automatisiert
  - Low-Usage Apps: viele Apps pro Host
  - High-Usage Apps: viele Hosts für eine App
- Zustandslose APIs simpel zu replizieren
- Memcache wird gespiegelt
- Datastore läuft auf Bigtable
  - Bigtable ebenfalls designet um gut zu skalieren
  - daher unterstützt Datastore-API keine Joins

- Schutz der Apps und des Systems
  - OS-Funktionalität eingeschrängt
- keine Unterprozesse / Threads / dynamische Libs
- keine Sockets / Ports
- keine Schreibzugriffe direkt aufs Dateisystem
  - jedoch auf Datastore
- verbot unsicherer Erweiterungen (zB Python ctypes)
- Limitierung der Ressourcen
  - 1000 Dateien pro App, je max 1 MB
  - Anfragen dauern max 30 sec und haben max 1MB

- bisher: Python 2.5.2
  - auf „Django“ aufsetzend
- Seit April 2009 auch Java 5 und 6
  - Groovy
  - Scala
  - JRuby
- Grails-Support angekündigt (Grails v1.1.1)
- SDK & Eclipse-Plugin

- native Python module (in C) sind verboten
- Viele Java Bibliotheken arbeiten nicht
  - whitelist
- Keine langlebigen Prozesse (max 30 sek)
- keine Forks / Threads / Unterprozesse
- CronJob-Intervall größer als eine Minute
- HTTPS nur zu XYZ.appspot.com
  - und nur mit Googles zertifikat
  - wegen Limitierungen des SSL-Protokols in Googles Cloud

Abbildung: iGoogle

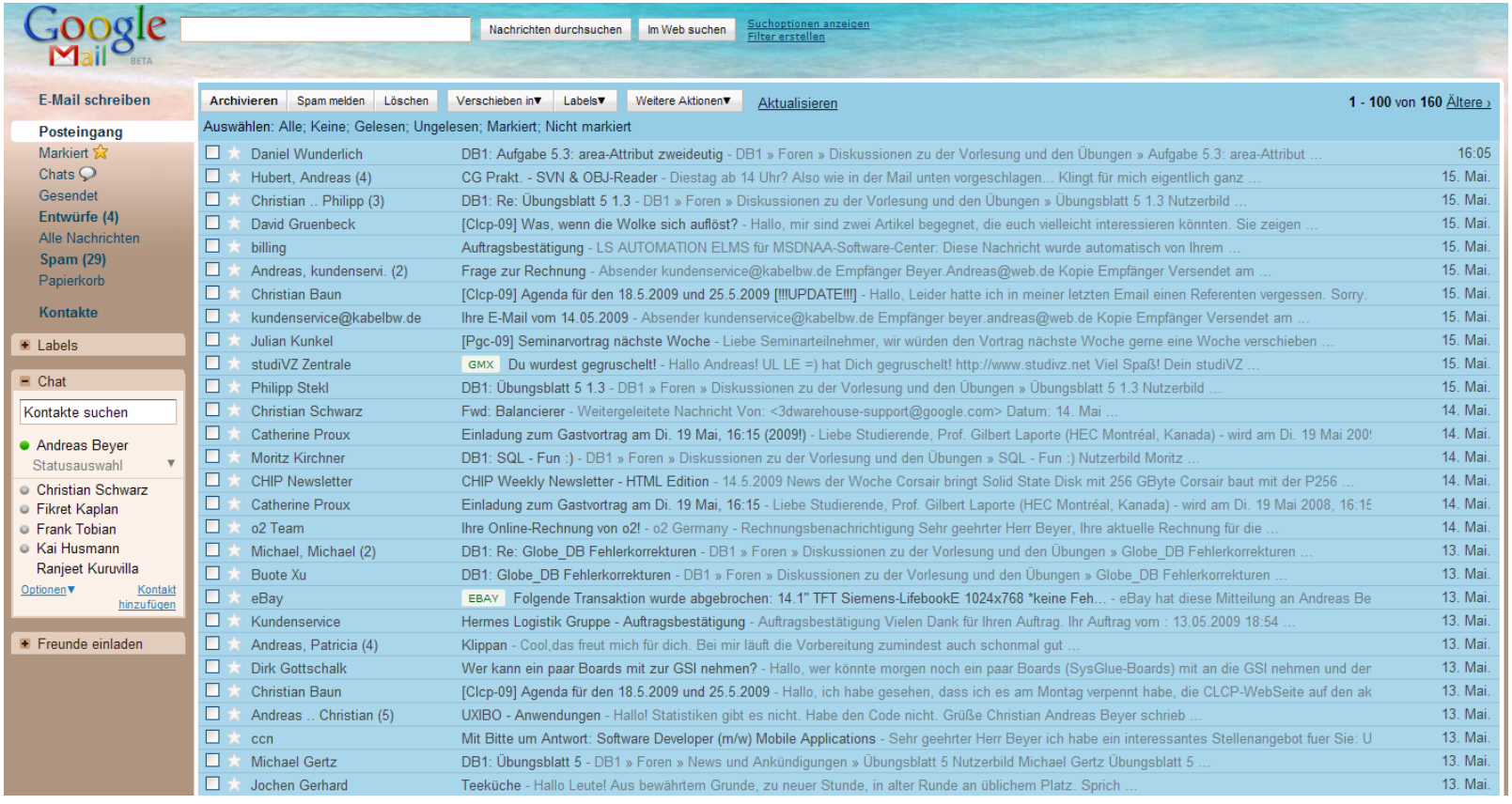

Abbildung: GMail

Christian Schwarz an mich — Details anzeigen 08:42 (Vor 22 Minuten) — Antworten

Titel: **Weinlagenwanderung**
Zeitpunkt: Fr 1. Mai 10:00 – 16:00 (CEST)
Wo: von Heppenheim nach Zwingenberg
Wer: Christian Schwarz*
Detaillierte Informationen »

Teilnehmen? Ja - Vielleicht - Nein

Ihr Tagesplan für Fr 1. Mai 2009

Keine früheren Veranstaltungen

10:00 **Weinlagenwanderung**

Keine späteren Veranstaltungen

Kalender anzeigen »

Google Kalender BETA

Andreas Beyer, Sie wurden eingeladen zu

Weinlagenwanderung

Fr 1. Mai 10:00 – 16:00
(Zeitzone: Berlin)
von Heppenheim nach Zwingenberg (Karte)
Kalender: Klausuren

Eigentümer/Veranstalter c.schwarz.mail@googlemail.com

Bahnhof/Haltestelle Datum Zeit Gleis Produkte
Heidelberg Hbf Fr, 01.05.09 ab 10:21 3 RB 15452
Heppenheim(Bergstr) Fr, 01.05.09 an 10:55 1

Das wäre die eine Alternative, wir werden vermutlich mit dem Auto nach Weinheim fahren, damit wir umziehsachen usw wegen Geburtstag von meinem Vater dabei haben.
Da könntet ihr mitfahren, wir steigen dann in diesen Zug in Weinheim dazu.

Bahnhof/Haltestelle Datum Zeit Gleis Produkte
Weinheim(Bergstr) Fr, 01.05.09 ab 10:45 2 RB 15452
Heppenheim(Bergstr) Fr, 01.05.09 an 10:55 1

Zurück gehts so um 16 Uhr.
Weitere Termindetails»

Nehmen Sie teil?
Ja | Vielleicht | Nein

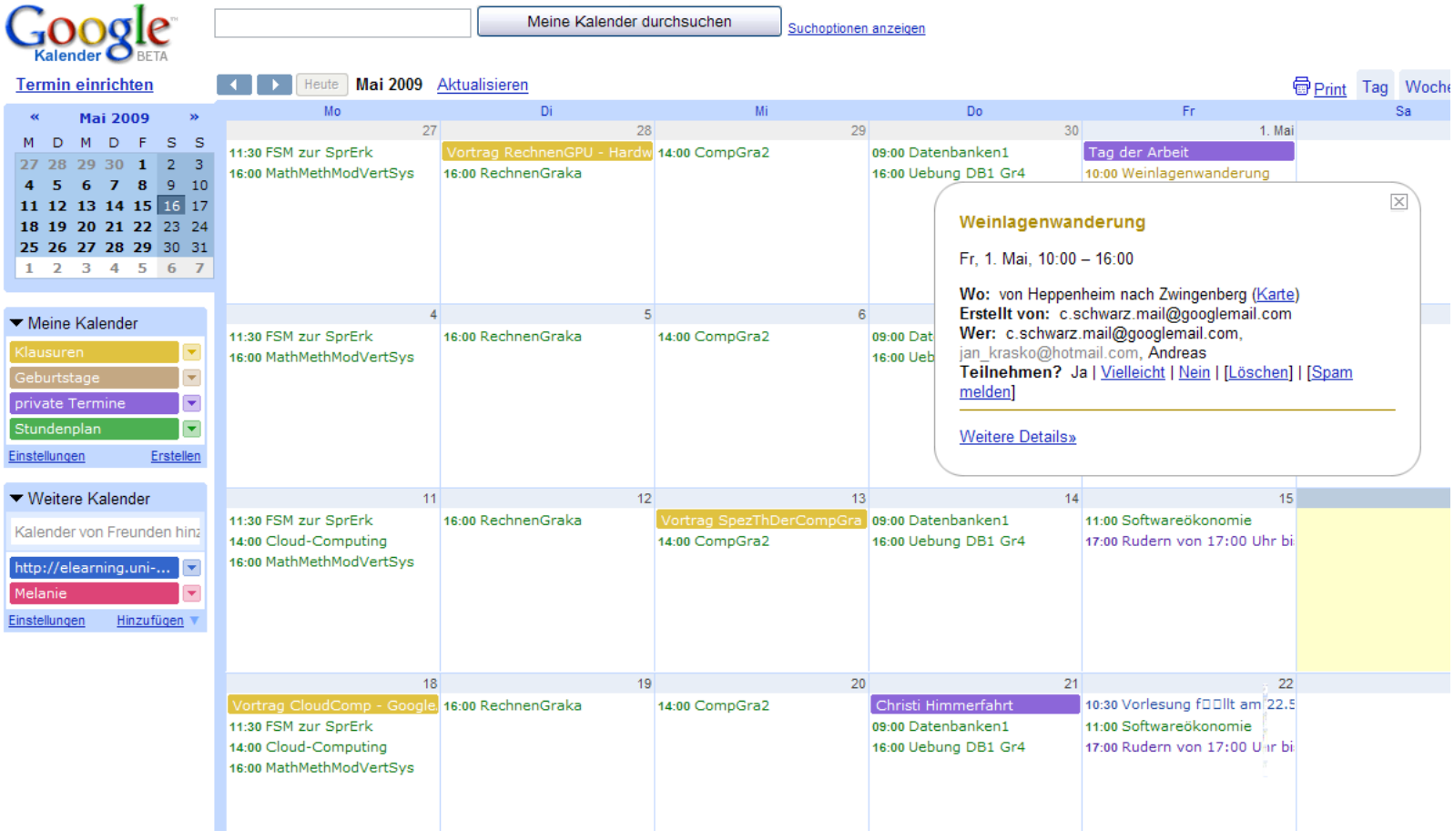

Abbildung: Google Kalender

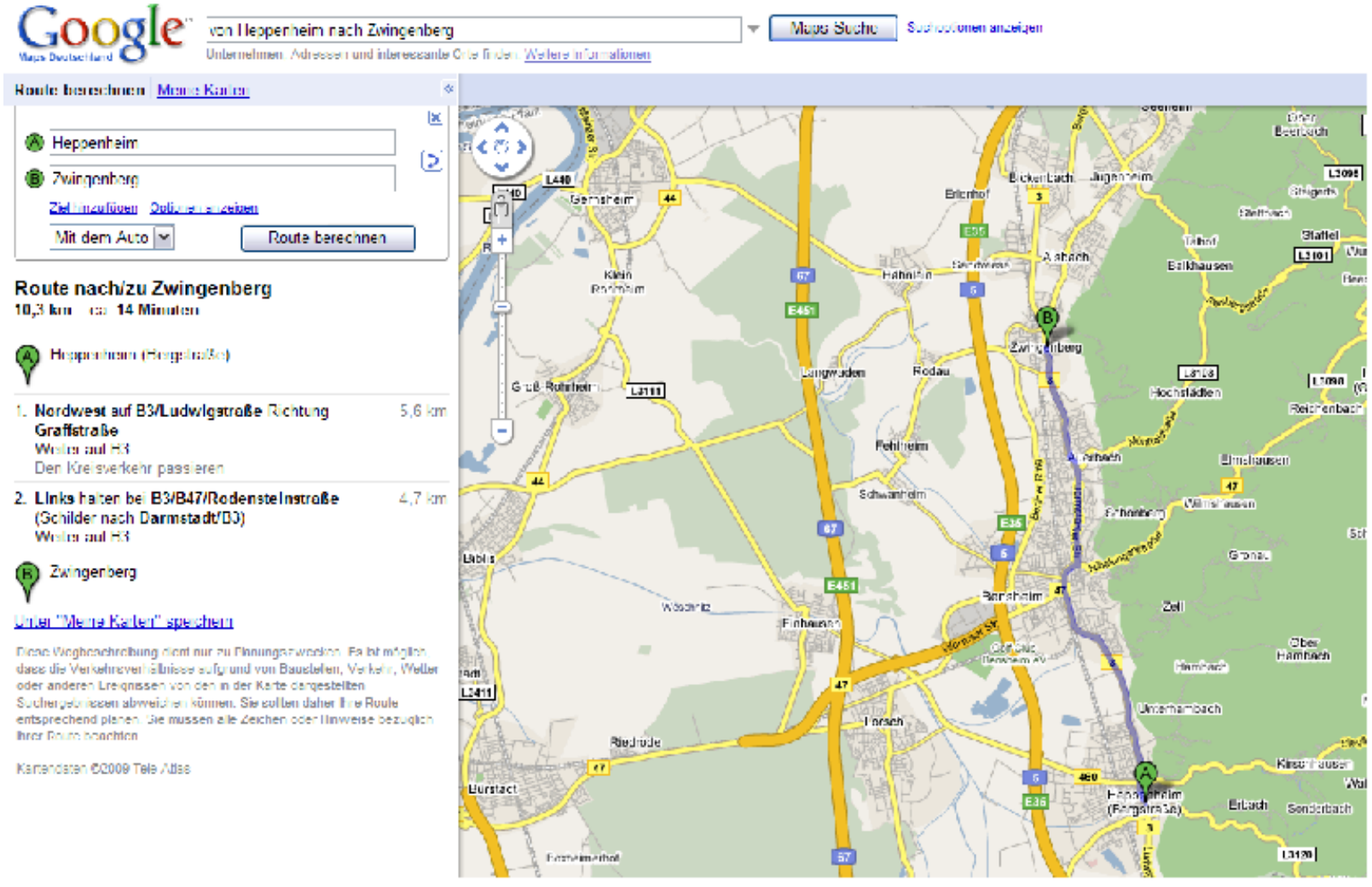

Abbildung: Google Maps

Abbildung: Google Docs

Google Text & Tabellen BETA

andreas.beyer.80@googlemail.com | Te

Steckbrief bearbeitet am 06.05.09 11:15 von Christian Schwarz

Freigeben | Speichern

Datei Bearbeiten Ansicht Einfügen Format Tabelle Tools Hilfe

Normal | Comic Sans M | 24 pt | B | I | U | A | Link

Steckbrief - Balancierer

- **Praktikanten:**

Christian Schwarz (Inf.)

Andreas Beyer (Inf.)

- **Betreuer:**
  - Benjamin Reh

- **Beschreibung:**
  - In unserem Praktikum „Balancierer" wird eine mechanische Wippe

balancierer

Mechanische Wippe bauen

Project hosting will be READ-ONLY Tuesday at 11am PDT due to brief network maintenance.

Project Home | Wiki | Issues | Source | Administer

Checkout | Browse | Changes | Search Trunk

How-to: Explore this project's source code by clicking the "Browse" and "Changes" links above. hide

Command-Line Access

If you plan to make changes, use this command to check out the code as yourself using HTTPS:

```
# Project members authenticate over HTTPS to allow committing changes.
svn checkout https://balancierer.googlecode.com/svn/trunk/ balancierer --username Andreas.Beyer.80
```

When prompted, enter your generated googlecode.com password.

Use this command to anonymously check out the latest project source code:

```
# Non-members may check out a read-only working copy anonymously over HTTP.
svn checkout http://balancierer.googlecode.com/svn/trunk/ balancierer-read-only
```

GUI and IDE Access

This project's Subversion repository may be accessed using many different client programs and plug-ins. See your client's documentation for more information.

This project is currently using approximately 0 bytes (0.0%) of its 1024 MB repository quota.

New project? You can reset this repository so that `svnsync` can be used to upload existing code history.

.

- Users API
- Images API
- URLFetch API
- Mail API
- Memcache API
- Datastore API

Users API

- Nutzer verifizieren über Google-Accounts
  - erspart das Programmierung eines Login
  - und das Schützen des Admin-Bereichs
- Admin hat Zugriff auf Nickname und Mail
- API stellt generierung von Buttons bereit
- muß nicht verwendet werden
  - und funktioniert nur mit Google-Accounts

.

Example

```
user = users.get_current_user()
if user: users.create_logout_url(self.request.path)
else: users.create_login_url(self.request.path)
```

Images API

- Biete Möglichkeiten Bilder einzubinden und zu manipulieren
  - resize()
  - crop()
  - rotate()
  - horizontal_flip() / vertical_flip()
- muß nicht verwendet werden

.

Example

```
photo = self.request.get("my_photo")
img = images.Image(photo.full_size_image)
img.resize(width=80, height=100)
img.im_feeling_lucky()
thumbnail = db.Blob(img.execute_transforms (
output_encoding=images.JPEG))
```

URLFetch API

- bietet die Möglichkeit Informationen aus anderen Seiten einzubinden
- unterstützt GET, POST, PUT, HEAD
- Kann der Anfrage eigene Daten beilegen
- Unterstützt Header für den Request
- es gibt keine andere Möglichkeit der Interaktion (Sockets / Ports)

.

Example

urlfetch.fetch(url, payload=None, method=GET, headers={}, allow_truncated=False)

URLFetch API - Antwort

enthält folgende Attribute

- headers
- status_code
- content
- content_was_truncated

Mail API

- Zum Versand von Mails
  - natürlich auch außerhalb Google
- alle gängigen Felder können frei belegt werden
- Ausnahme:
  - Sender ist „current_user“ oder AppAdmin
- es gibt keine andere Möglichkeit zu Senden

Mail API - Quota

freie Quota kostenpflichtige Quota Tageslimit Maximalrate Tageslimit Maximalrate Mail API Aufrufe 7000 Aufrufe 32 Aufrufe/Minute 4900 Aufrufe/Minute an Empfänger 2000 Empfänger 8 Empfänger/Minute 5100 Empfänger/Minute 5000 Mails 24 Mails/Minute 9700 Mails/Minute 60 MB 29 GB 84 MB/Minute Anzahl Anhänge 2000 Anhänge 8 Anhänge/Minute 8100 Anhänge/Minute 100 MB 100 GB 300 MB/Minute Resource 1,7 Mio Aufrufe max 7,4 Mio Empfänger an Admin selbst 3 Mio Mails Message Body Daten 340 kB/Minute 2,9 Mio Anhänge größe der Anhänge 560 kB/Minute

Memcache API

- wie der Name schon sagt - ein Cache
  - hält vorgeladene Anfragen an Datastore
- legt Ergebnisse in RAM der Maschine(n) auf der App ausgeführt wird
- beschleunigt Anfragen
- muß nicht benutzt werden

.

Example

```
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
```

Datastore API

- setzt auf BigTable auf
- Nutzt schemenfreie GQL-Syntax
- erlaubt Ablage persistenter Daten zwischen Aufrufen der Applikation
- verwaltet Datenobjekte als „Entities“
  - diese haben „properties“ und „values“
  - können zur Optimierung indiziert werden
- sichert Konsistenz, verbietet Joins
- sonst keine andere Art der Speicherung

.

Example

```
class Shout(db.Model):
message = db.StringProperty(required=True)
when = db.DateTimeProperty(auto_now_add=True)
... shout = Shout(message=self.request.get('message'))
shout.put()
... shouts = db.GqlQuery('SELECT * FROM Shout' 'ORDER BY
when DESC').fetch(100)
```

.

- ermöglicht zyklische Ausführung von Befehlen
- gleiche Syntax wie UNIX-Systeme
- definiert in cron.yaml
- erlaubt Intervalle von minimal einer Minute
- Google bietet keine andere Möglichkeit an
  - Threads etc. sind verboten

Alternativen

- Anstoß durch extern laufende Dienste
  - zum Beispiel webcron.org (0,0001€ pro „hit“)
- Anstoß durch CronJob auf eigenem PC
- Googles Cloud und Amazon-Cloud verbinden
  - GoogleAppEngine unterstützt „Boto“
  - „Boto“ ist Python-Erweiterung für Amazon
  - mittels Boto Aufgaben in Amazon-SQS einreihen
  - 0,01$ pro 10000 Anfragen (+ 0,10$ pro GB)

.

Example

Crontab analog zu Unix

```
* * * * * auszuführender Befehl
          —- Wochentag (0-7) (Sonntag =0 oder =7)
        —— Monat (1-12)
      ——— Tag (1-31)
    ———- Stunde (0-23)
  ——————— Minute (0-59)
```

Bsp: „0 17 * * 1-5 /bin/echo Feierabend“

.

- Datenbank
  - komprimiert
  - proprietär
  - relational (also inkl Joins)
- Ähnelt sortiertem Hashtable
- Setzt auf GoogleFS auf
- Nutzt GQL-Syntax, „angelehnt“ an SQL

.

- Schemenfreies Datenbanksystem
- skaliert gut auf BigTable
- biete Transaktionen, Anfragen, Suchen
- keine Joins
  - rechenintensiv und strukturändernd

- Google erlaubt Untermenge von Java
  - keine Threads
  - keine Dateisystemzugriffe
- Kommunikation mit Datastore über APIs
  - Java Data Objects (JDO) 2.3
  - Java Persistence API (JPA) 1.0
- Kommunikation mit MemCache über API
  - JCache (JSR 107)
- ebenso MailAPI über JMail

Ruby (JRuby)

- Objektnamen definieren Zusammenspiel
- ermöglicht „agile Softwareentwicklung“
  - Prinzipien „Don’t Repeat Yourself“ (DRY)
  - „Convention over Configuration“
- Ruby-on-Rails ist quelloffenes Web Application Framework
- JRuby ist Implementierung eines Ruby- Interpreters in Java

Scala

- funktionale und objektorientierte Programmiersprache
- erleichtert parallele Programmierung für Multikernprozessor-Systeme
- nahezu beliebige Bezeichner für Methoden
- sehr mächtiges Typensystem
- realisiert eine Art Mehrfachvererbung in Java

Groovy

- dynamisch typisierte Programmiersprache & Skriptsprache
- versuch Java-Syntax mit Konzepten von Ruby zu verbinden
- erweitert Java um
  - Closures
  - native Syntax für Maps, Listen und Reguläre Ausdrücke
  - ein einfaches Templatesystem, mit dem HTML und SQL-Code erzeugt werden kann
  - eine XQuery-ähnliche Syntax zum Ablaufen von Objektbäumen
  - Operatorüberladung
  - native Darstellung für BigDecimal und BigInteger
- wegen einfachen Handhabung von BigDecimal unter anderem im Finanzbereich eingesetzt

Grails

- wie Ruby-on-Rails für Ruby
  - Groovy-on-Rails
- also quelloffenes Web Application Framework

Django

- Python Web Framework
  - bildet Gegenstück zu „Ruby-on-Rails“
- allerdings mit expliziter Konfiguration
- Abbildung einer URL auf eine Python-Funktion
  - folgt Model-View-Controller-Schema
- Datenmodell / Präsentation / Programmsteuerung

Python

- einfache/übersichtliche Programmiersprache
  - reduzierte Syntax / wenige Schlüsselwörte
  - Interpretersprache von 1991
  - mehrere Paradigmen unterstützt
- objektorientiert
- aspektorientiert
- funktional

- kostenlos:
  - 500 MB pro Anwendung
  - 5 Millionen Seitenaufrufe pro Seite
  - 10 Anwendungen pro Nutzer
- zukaufbar (fällt bei Überschreitung an):
  - Traffic (In: 0,10$ pro GB / Out: 0,12$ pro GB)
  - CPU Zeit (0,10$ pro CPU-Stunde)
  - Datastore-Speicher (0,15$ pro GB pro Monat)
  - ab 20.001er Mail (0,0001$ pro Empfänger)

- Entwickeln einer einfachen Applikation
- Test im SDK
- „Deploy“ zu Google
- Ausführung im Browser
- Ansicht der Admin-Console
- Ansicht des Bezahl-Protals

- Google bietet einfachen und (begrenzt) kostenlosen Rahmen zum Anbieten von WebApplikationen
- bisherige Grundsprachen: Python und Java
- Sandbox isoliert Anwendung von System – Kritik: Spielwiese kann zu Käfig werden
- stellt Plattform und eigene Dienste / APIs
- löst Probleme der Skalierung und Ressourcen (teils gegen Bares)

- www.Google.com
  - code.google.com/intl/de/appengine
- www.Wikipedia.de
  - Ruby / JRuby
  - Groovy / Grails
  - Scala
  - Python
  - Django