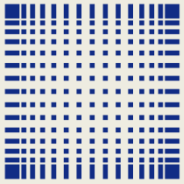


hochschule mannheim

Konzeption und Entwicklung einer Cloud-basierten, redundanten Speicherlösung

Kolloquium zur Bachelorthesis

Melchior Jurczyk

19.04.2011

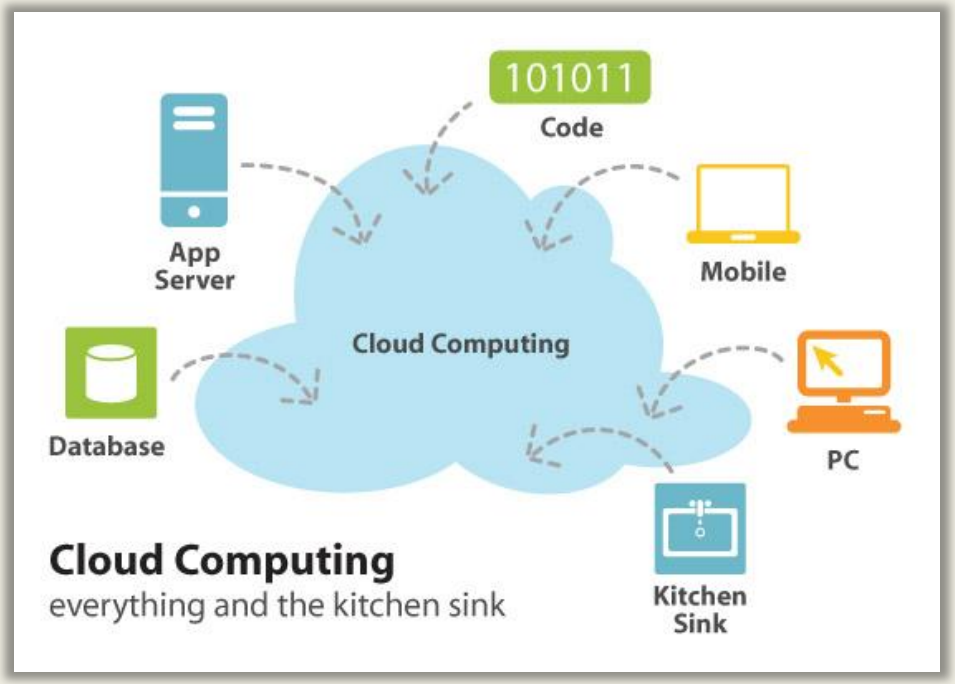
Agenda

- Aufgabenstellung
- Hintergrund
- Konzeption
- Verwendete Dienste
- Eckdaten
- Implementierungsdetails
- Entscheidungen
- Vorstellung der Anwendung
- Vor- & Nachteile
- Zusammenfassung & Ausblick

Aufgabenstellung

- Motivation
  - Datensicherungen sind wichtig
  - Lokale Aufbewahrung der Sicherungen ist unsicher
  - Entfernte Aufbewahrung ist etwas sicherer
  - Auf Datenhaltung spezialisierte Anbieter ermöglichen einen hohen Grad an Sicherheit
- Aufgabenstellung
  - Konzeption und Entwicklung einer Cloud-basierten, redundanten Speicherlösung

Hintergrund: Cloud Computing



Quelle: www.huffingtonpost.com

Eigenschaften:

-Verbrauchsabhängige Abrechnung
-Nutzung auf Abruf (on-demand)
-Ortsunabhängige Verwendung
-Benutzerfreundlich
-Flexibel belastbar (Skalierung)
-Virtualisierte Ressourcen
-Kostensenkung für Kunden & Anbieter

Konzeption

- Cloud-basiert
  - Wird in der Cloud betrieben und nutzt Cloud-Speicherdienstanbieter
- Redundanz
  - Ausfall einer Kopie stellt keinen Totalausfall dar
- Anbieterunabhängig
  - Um auf einzelne Anbieter nicht angewiesen zu sein
- Transparenz der Verteilung
  - Automatisierung zur Vereinfachung der Bedienung
- Erweiterbar
  - Anbieter können flexibel ausgetauscht werden

Verwendete Dienste

- Amazon Simple Storage Service (S3)
  - Speicherdienst der Amazon Web Services
  - Als „Speicher für das Internet“ konzipiert
  - Key/Value basierte Ablage der Daten
- Google Storage
  - Key/Value basierte Ablage der Daten
  - Kompatibel zu S3
- Google App Engine
  - Laufzeitumgebung zum Betreiben von Webanwendungen
  - Alle notwendigen Ressourcen werden bereit gestellt

Eckdaten

- Lösung als Dienst
  - Keine Installation oder Updates
  - Kein Wartungsaufwand
  - Ortsunabhängige Nutzung
- Trennung von Upload und Verteilung
  - Abarbeitung im Hintergrund
  - Erweiterbarkeit
  - Loskopplung von GUI-Interaktion

Implementierungsdetails

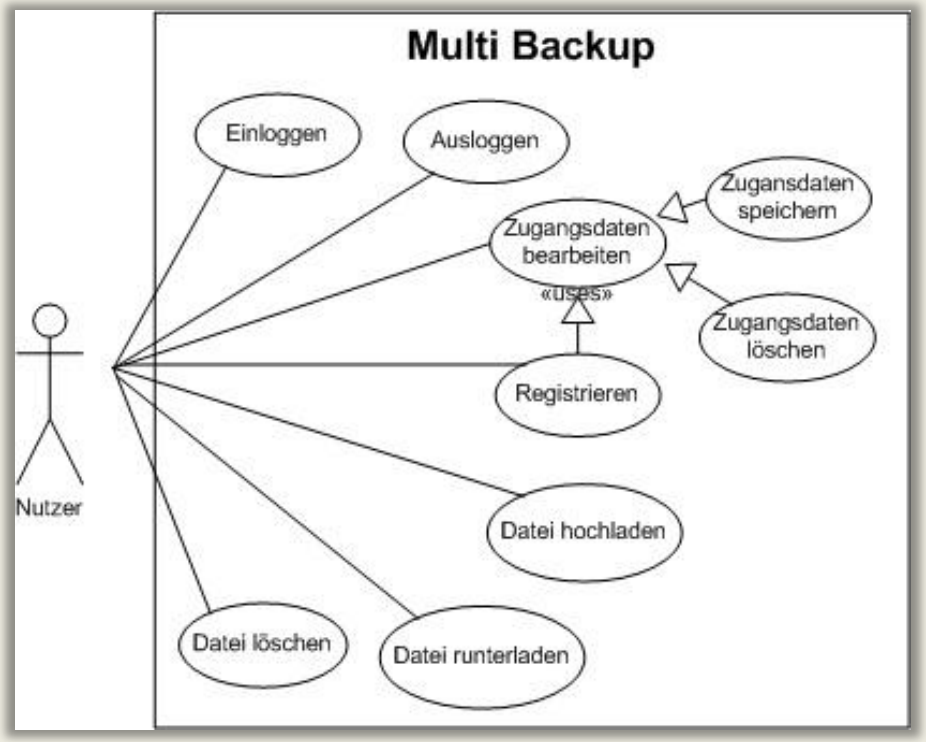
- MVC
  - Austauschbarkeit der GUI und strukturierter Quellcode
- Boto Interface
  - API für S3 und dazu kompatibler Speicherdienste
- Django Templates
  - Dynamische Generierung von HTML-Seiten
- App Engine Laufzeitumgebung
  - Max. Request-Dauer auf Response 30 Sek.
  - Einschränkungen der verwendeten APIs

Entscheidungen

- Upload
  - → Direkt oder indirekt?
- Blobstore Up- & Download
  - → Javascript oder HTML Schachtelung?
  - → URL-Fetch oder Blobstore.Reader?
- Django Renderer
  - → Übersichtlichkeit dank Tag-System
- 30 Sekunden Beschränkung
  - → Task API oder Deferred API?

Vorstellung der Anwendung

- MultiBackup
  - Einloggen
  - Zugangsdaten eingeben
  - Datei
    - Hochladen
    - Runterladen
    - Löschen
  - Ausloggen



Vor- & Nachteile

- Vorteile
  - Speicherdienstunabhängigkeit
  - Redundante Datenhaltung
  - Erweiterbarkeit um weitere Dienste
  - Datenzugriff auch ohne MultiBackup möglich
  - Mehrere Dienste über eine Anwendung nutzbar

- Nachteile
  - Vertrauen gegenüber:
    - MultiBackup (bzgl. Zugangsdaten)
    - Speicherdienste (bzgl. Daten)
  - Mehrere Nutzerkonten & Abrechnungen
  - Uneingeschränkte Dateigröße nur in privaten Infrastrukturen

Zusammenfassung & Ausblick

- Zusammenfassung
  - Anforderungen erfüllt
  - Trotz Einschränkungen Verwendung der App Engine vorteilhaft
  - App Engine nicht optimal für datenintensive Anwendungen
- Ausblick
  - Upload mehrerer Dateien gleichzeitig
  - Integration zusätzlicher Anbieter
  - Lokaler Client für automatisierte Backups/Synchronisation
  - Verschlüsselung der Daten und des Transports

Fragen?

Vielen Dank

für Ihre

Aufmerksamkeit!

OFFENE DISKUSSION ERWÜNSCHT ☺