Google App Engine

Alicja Łuszczak

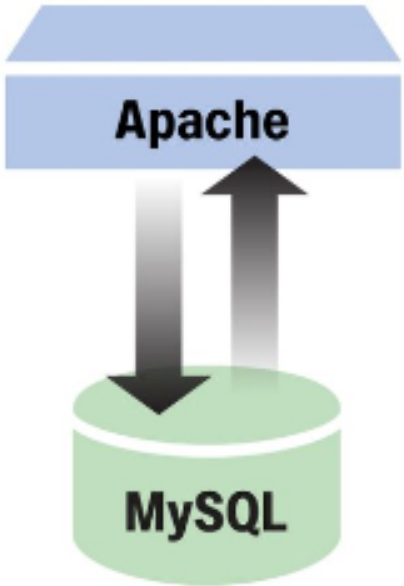
8 stycznia 2010

Plan prezentacji

Google App Engine

- Platforma cloud computing firmy Google.
- Do tworzenia i hostowania aplikacji webowych.
- Wersja beta – kwiecień 2008.

Jak działają początkujące aplikacje webowe?

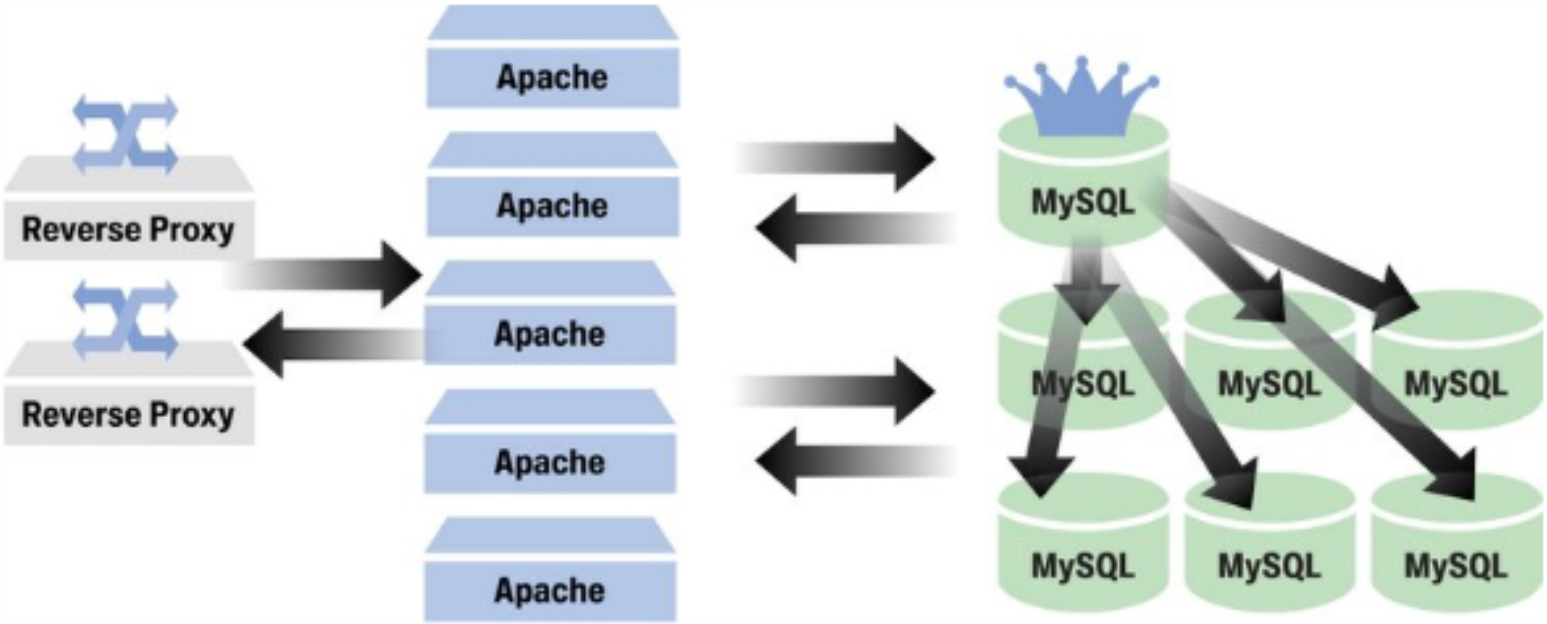


Serwer Apache

+

Serwer MySQL

A jak działają kiedy zdobędą popularność?



- Równoważenie obciążenia serwerów z Apachem.
- Podzielona baza danych.

Problemy

- Mnóstwo pracy przy zarządzaniu:
  - utrzymywanie dużej liczby maszyn,
  - konfiguracja (system operacyjny, oprogramowanie),
  - równoważenie obciążenia,
  - wykrywanie awarii,
  - tworzenie kopii zapasowych, itp.
- Skomplikowane korzystanie z bazy danych.

Dalsze problemy

- Konieczność przepisywania kodu aplikacji w miarę, jak wzrasta popularność.
- Duże prawdopodobieństwo, że rozwój aplikacji nie nadąży za wzrostem obciążenia.
- W przypadku spadku popularności, zostajemy z dużą ilością zbędnego sprzętu.

A rozwiązanie to...

Najważniejsze idee

- Aplikacja jest odizolowana od sprzętu, systemu operacyjnego i innych aplikacji, przez środowisko, w którym jest uruchamiana.
- Dla programisty liczy się tylko specyfikacja środowiska, jakie dostarcza GAE.

Najważniejsze idee

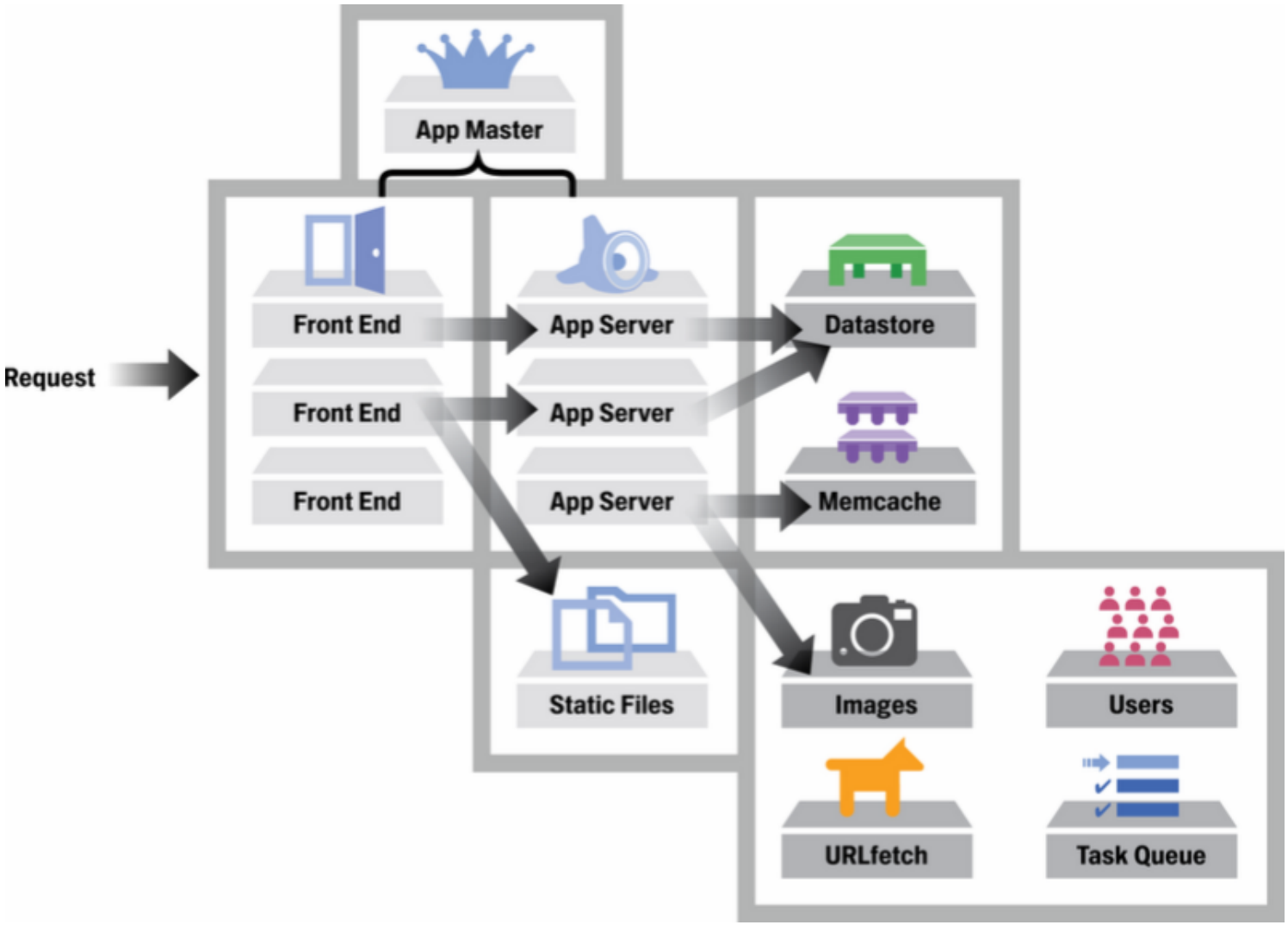
- Kod aplikacji wykonywany jest jedynie jako reakcja na request:
  - wejście na stronę aplikacji,
  - przychodzący e-mail,
  - wiadomość na czacie,
  - zadanie z kolejki,
  - zaplanowane zadanie.
- Każdy request może być przetworzony na innej maszynie.
  - brak stanowości,
  - brak dostępu do sprzętu.

Najważniejsze idee

- Wiele aplikacji jednocześnie współdzieli te same maszyny.
- GAE zajmuje się:
  - równoważeniem obciążenia maszyn,
  - wykrywaniem awarii,
  - tworzeniem kopii bezpieczeństwa,
  - zwielokrotnieniem często używanych danych,
  - zarządzaniem rozproszonym cachem oraz bazą danych.

Aplikacja jest całkowicie zwolniona z odpowiedzialności za nie.

Architektura



Front-endy

- Na podstawie URLa dzielą requesty na:
  - dot. elementów statycznych (np. CSSów, obrazków), które samodzielnie przesyłają,
  - dot. elementów dynamicznych, które przekazują do serwerów aplikacji.
- Potrafią rozpoznawać zalogowanych użytkowników Google i tylko im przekazywać odpowiednie treści.
- Odpowiadają za kompresję (gzip).
- Ich obciążenie jest automatycznie równoważone.

App Master

- Przekazuje nową wersję aplikacji
  - front-endom,
  - serwerom aplikacji,
  - serwerom plików statycznych,

  w taki sposób, by nie spowodować kolizji ze starą wersją.

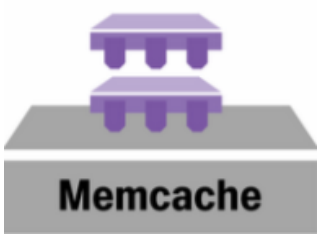
Serwery aplikacji

- Wiele aplikacji na jednym serwerze.
- Wiele requestów przetwarzanych równolegle.
- Cache'ują środowiska wykonania aplikacji do ponownego użycia.
- Automatycznie równoważą obciążenia na podstawie charakterystyki danej aplikacji.

Datastore

- To coś w rodzaju jednej, olbrzymiej, rozproszonej, posortowanej tablicy.
- Pozwala na bardzo szybkie odczyty nawet przy bardzo dużej ilości danych.
- Wykorzystuje GQL, podzbiór SQL.
- Z pewności *nie jest* relacyjną bazą danych.

Memcache



- Wydajny, rozproszony cache na obiekty.
- Przeznaczony przede wszystkim dla uniknięcia wielokrotnego pobierania tych samych danych z Datastore.

- Pozwala na pobranie danych spod dowolnego adresu URL.
- Protokołów HTTP lub HTTPS.
- Umożliwia korzystanie z zewnętrznych usług.

- Wysyła wiadomości (z adresu administratora lub z adresu zalogowane użytkownika).
- Odbiera maile.
- Obsługuje załączniki.

- Odbieranie i wysyłanie wiadomości, zapraszanie do chatu dowolnego użytkownika.
- Np. „chat bot”, powiadomienia.

- Pozwala na dodawanie zadań do wykonania „w tle”.
- Polega na odpaleniu przez App Engine handlera przypisanego do odpowiedniego adresu URL.

- Pozwala na manipulowanie obrazami: przycinanie, skalowanie, konwersję, „ulepszanie”, uzyskanie informacji o obrazie.

- Umożliwia wykrycie zalogowania do konta Google.
- Daje dostęp do adresu e-mail użytkownika.
- Pozwala na odróżnienie administratorów strony.

Blobstore

- Pozwala na zapisywanie plików nadesłanych przez użytkowników.
- Służy do przechowywania danych o dużym rozmiarze, które nie nadają się do umieszczenia w Datastore.
- Wprowadzone w grudniu 2009.

Więcej o Datastore

Ograniczenia Datastore

- Każdy obiekt zapisywany w Datastore musi mieć swój *rodzaj* (kind).
  - Jednym zapytaniem możemy pobierać tylko obiekty jednego rodzaju.
  - Obiekty jednego rodzaju nie muszą mieć tych samych właściwości.
- $<, \leqslant, \geqslant, <$ tylko na jednej właściwości w danym zapytaniu.
- Jednym zapytaniem możemy odczytać do 1000 obiektów.
- Wydajność jest uzależniona od odpowiedniego doboru indeksów.

GQL

```
SELECT [* | __key__] FROM <kind>
    [WHERE <condition> [AND <condition> ...]]
    [ORDER BY <property> [ASC | DESC]
    [, <property> [ASC | DESC] ...]]
    [LIMIT [<offset>,]<count>]
    [OFFSET <offset>]

<condition> := <property> {< | <= | > | >= | = | != }
                          <value>
<condition> := <property> IN <list>
<condition> := ANCESTOR IS <entity or key>
```

Transakcje w Datastore

- Działają w obrębie jednej grupy obiektów.
- Grupa obiektu jest określana w momencie utworzenia i nie może być potem zmieniona.
- Obiekty z jednej grupy nie muszą być tego samego rodzaju.
- Dla zwiększenia wydajności obiekty z jednej grupy są przechowywane w sąsiednich wierszach.

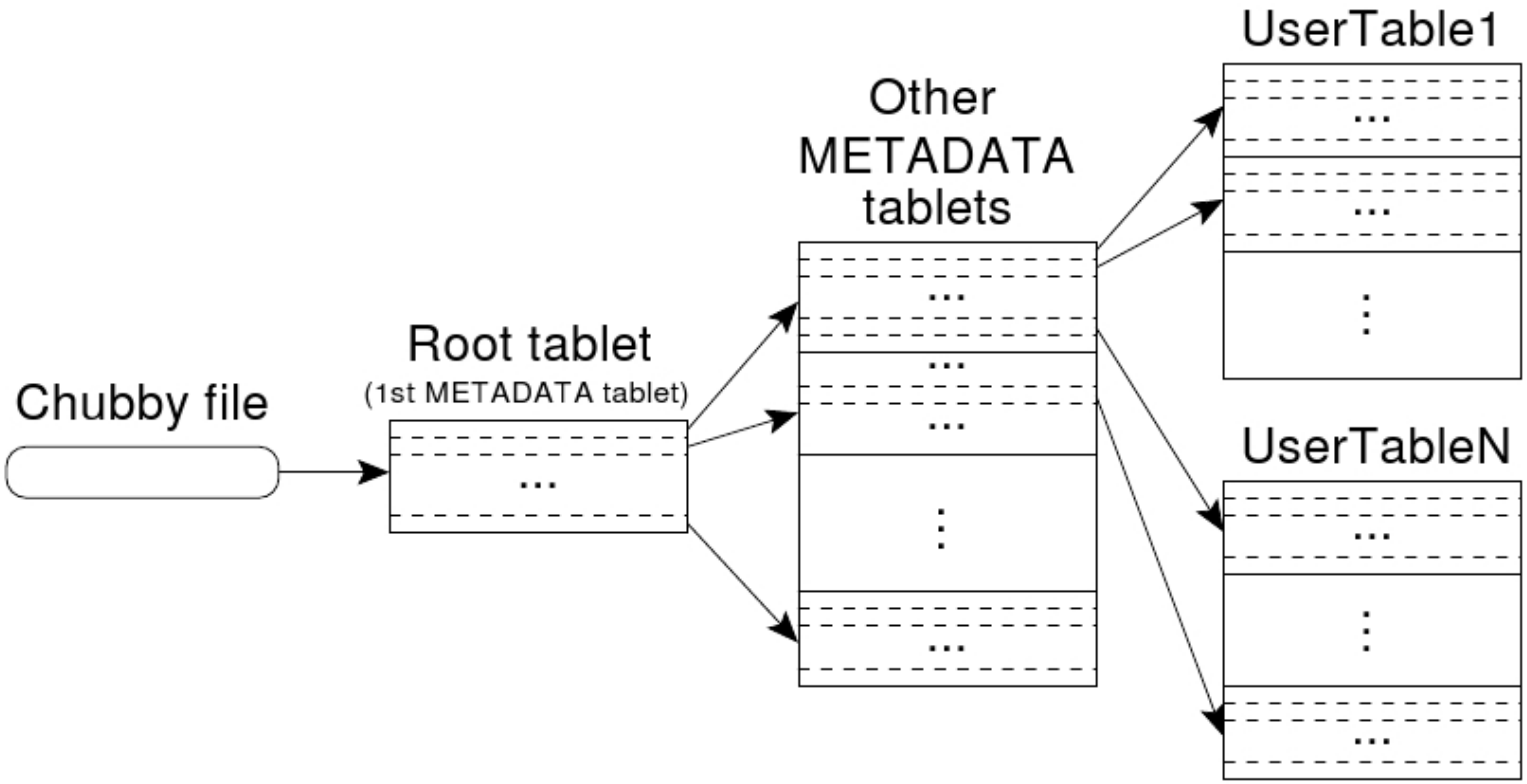
BigTable

- Datastore zbudowany jest na bazie BigTable.
- BigTable to rozproszony system zarządzania bazą danych zaprojektowany z myślą o ogromnych zbiorach informacji.
- BigTable jest wykorzystywany do przechowywania danych w:
  - Google Analytics,
  - Google Earth,
  - Orkut,
  - itp.

Jak działa BigTable?

- Dane podzielone są na *tablety*, które odpowiadają pewnym przedziałom kluczy.
- Tablety mają ok. 100 MB. Po przekroczeniu rozmiaru 200 MB zostają rozbite na dwa mniejsze tablety.
- Każdy tablet znajduje się w pamięci pewnego *serwera tabletów*.
- Tablety tworzą drzewo o wysokości 3.
  - Dwa pierwsze poziomy przechowują metadane.
  - Trzeci poziom przechowuje rzeczywistą zawartość tabeli.

Jak działa BigTable?



Jak działa BigTable?

- Lokalizacja korzenia drzewa tabletów przechowywana jest w odpowiednim pliku Chubby'iego.
- Za pierwszym razem klient musi przejść całe drzewo, aby ustalić adres potrzebnego mu serwera tabletów.
- Przy kolejnych użyciach bazy danych klient korzysta ze wcześniej spamiętanych danych.
- Informacje dotyczące odczytu i zapisu przesyłane są bezpośrednio między klientem a odpowiednim serwerem tabletów.

Programowanie w Google App Engine

Ograniczenia środowiska

- Aplikacja może czytać z systemu plików, ale nie może zapisywać.
- Dostęp do sieci tylko za pomocą odpowiednich usług (FetchURL).
- Nie ma żadnej gwarancji zachowania środowiska pomiędzy requestami.
- Nie ma pewności, że requesty od jednego użytkownika będą trafiały do tego samego serwera aplikacji.

Wykonanie naszego kodu

- Kod naszej aplikacji jest wykonywany jedynie jako obsługa requesta.
- Zawsze musi zakończyć działanie w ciągu 30 sekund albo jego wykonanie zostanie przerwane.

Języki programowania

- Można wykorzystywać jedynie kod w Pythonie i Javie.
- Można używać z dodatkowych bibliotek, o ile posiadamy ich kod źródłowy w Pythonie lub Javie.
- Można korzystać z frameworków webowych, np. Django, CherryPy, Pylons i web2py.

Dostępne narzędzia

- SDK:
  - deweloperski serwer webowy, symulujący środowisko GAE na naszym komputerze,
  - narzędzie do przesyłania aplikacji na serwer,
  - narzędzie do umieszczenia w Datastore danych np. z arkusza CSV.
- Dla Pythona:
  - lekki framework webapp, oparty na Django.
- Dla Javy:
  - wtyczka do Eclipse,
  - Google Web Toolkit.

Logi

- App Engine automatycznie zapisuje wszystkie requesty i podstawowe statystyki o nich.
- Możemy dodawać do nich własne informacje.

```
import logging

# ...
        logging.debug('debug level')
        logging.info('info level')
        logging.warning('warning level')
        logging.error('error level')
        logging.critical('critical level')
```

Konfiguracja aplikacji

app.yaml

```
application: ae-book
version: 1
runtime: python
api_version: 1

handlers:
- url: /css
  static_dir: css

- url: /.*
  script: main.py
```

Dostęp do Datastore

```
from google.appengine.ext import db
import datetime

class Song(db.Expando):
    title = db.StringProperty()

crazy = Song(title='Crazy like a diamond',
             author='Lucy Sky',
             publish_date='yesterday',
             rating=5.0)

crazy.put()
```

Dostęp do Datastore

```
from google.appengine.ext import db

class Greeting(db.Model):
    author = db.UserProperty()
    content = db.StringProperty(multiline=True)
    date = db.DateTimeProperty(auto_now_add=True)

greetings = db.GqlQuery("SELECT * FROM Greeting "
    + "ORDER BY date DESC LIMIT 10")

query = Greeting.all()
query.filter('author = ',author).order('-date')
results = query.fetch(10)
```

Darmowa quota

Resource	Free Default Quota		Billing Enabled Default Quota	
	Daily Limit	Maximum Rate	Daily Limit	Maximum Rate
Requests	1,300,000 requests	7,400 requests/minute	43,000,000 requests	30,000 requests/minute
Outgoing Bandwidth (billable, includes HTTPS)	1 gigabyte	56 megabytes/minute	1 gigabyte free; 1,046 gigabytes maximum	10 gigabytes/minute
Incoming Bandwidth (billable, includes HTTPS)	1 gigabyte	56 megabytes/minute	1 gigabyte free; 1,046 gigabytes maximum	10 gigabytes/minute
CPU Time (billable)	6.5 CPU-hours	15 CPU-minutes/minute	6.5 CPU-hours free; 1,729 CPU-hours maximum	72 CPU-minutes/minute

Resource	Free Default Quota	Billing Enabled Default Quota
Stored Data (billable)	1 gigabyte	1 gigabyte free; no maximum

Opłaty

Resource	Unit	Unit cost
Outgoing Bandwidth	gigabytes	$0.12
Incoming Bandwidth	gigabytes	$0.10
CPU Time	CPU hours	$0.10
Stored Data	gigabytes per month	$0.15
Recipients Emailed	recipients	$0.0001

Podsumowanie

Zalety:

- Pisanie skalowalnych aplikacji dla GAE jest łatwe.
- Pozwala nam sprawdzić, czy aplikacja „wypali”, nim zainwestujemy w nią pieniądze.

Wady:

- GAE nie nadaje się dla aplikacji, które wymagają relacyjnej bazy danych.
- Użycie GAE może grozić zamknięciem się w jednej technologii.

Pytania?

Bibliografia

- Levi, Alon. *From Spark Plug to Drive Train: Life of an App Engine Request.*
- Sanderson, Dan. *Programming Google App Engine.* Sebastopol: O'Reilly Media, 2009.
- *Bigtable: A Distributed Storage System for Structured Data.* http://labs.google.com/papers/bigtable-osdi06.pdf.
- http://code.google.com/appengine/
- Anikiej, Kamil. *SZBD Big Table.*