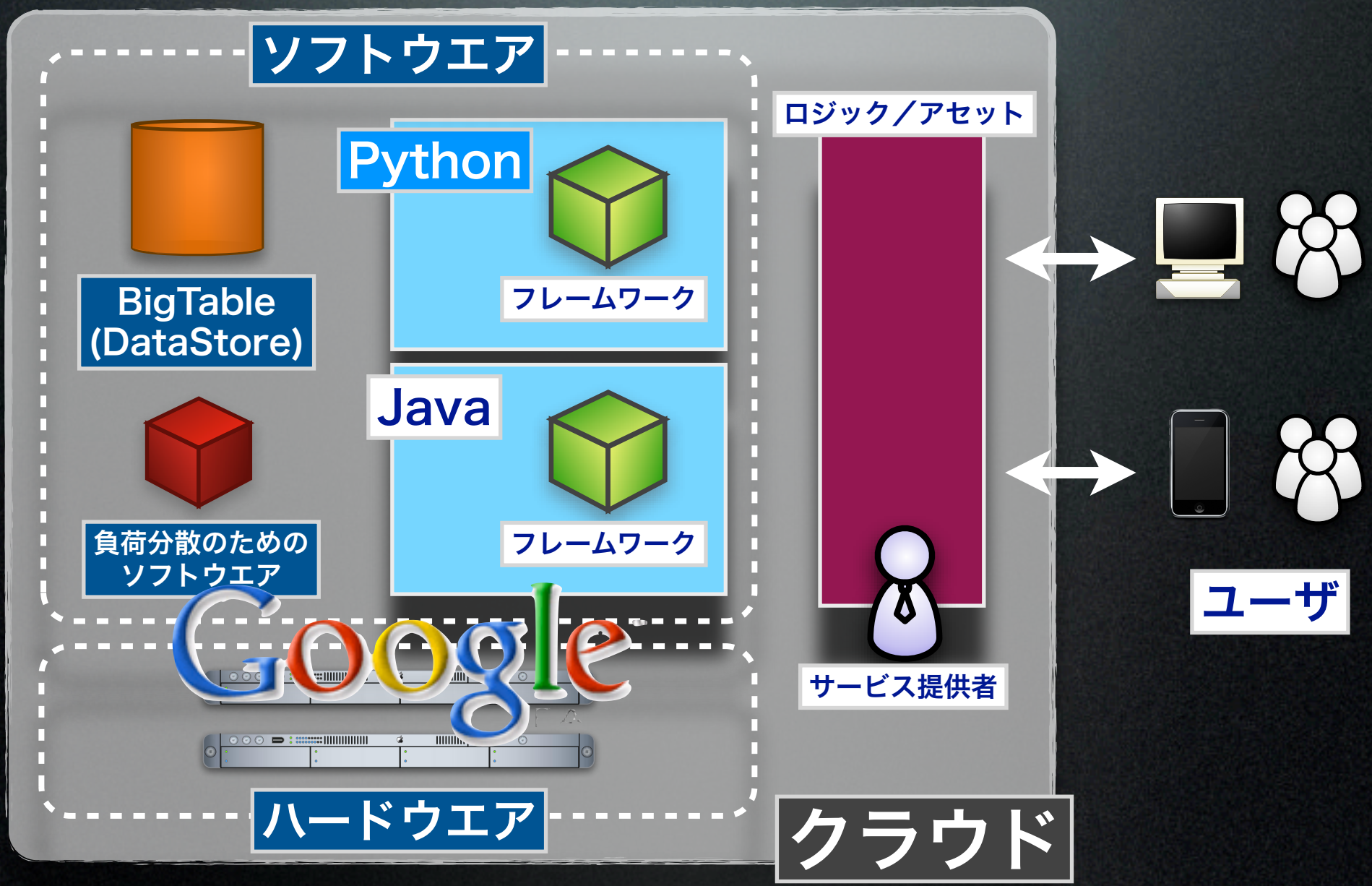
事例で見る Google App Engine の魅力

(c) ウェブコア株式会社 : 柴田 淳

Google App Engineとは？

- Web上のサービスを作るための環境
  - Webアプリケーション
  - Webサービス etc
- Googleのインフラを利用
  - スケーラブル
  - チープ

PaaSとしてのApp Engine



事例

Google Moderator

- App Engine製の「Web投票」を簡易に設置できる集合知アプリ

Google Moderator (2)

- 2009年3月26日，オバマ大統領がオンライン討論会の質問を決めるのにModeratorを利用

- 数時間で7000の質問，24万の投票
- App Engineのスケーラビリティを証明

Google 未来を選ぼう 衆院選 2009

- 2009年衆院選Googleキャンペーンサイト

Google 未来を選ぼう 衆院選 2009(2)

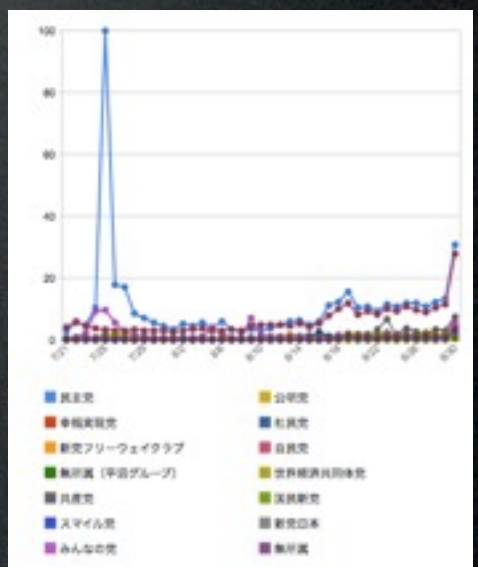
- バックエンドはApp Engine(弊社担当)
  - 立候補者，選挙結果，検索数，投票所などのデータを動的に配信
- Google Mapsとのマッシュアップ
  - Mapsを使った選挙区検索，投票所検索

案件の性質(2)

- 多くのアクセス
  - Googleトップからのリンク
  - 検索ページからの導入
  - 政権交代がかかる話題性の高い選挙

案件の性質(2)

- 短納期
- 複雑な要件
  - 大量のデータを扱う
  - Google Mapsなどフロントエンドとの連携
- アジャイルな開発



App Engineの利点を活用

- スケーラビリティ
- 運用の手軽さ
- チープ

スケーラビリティ

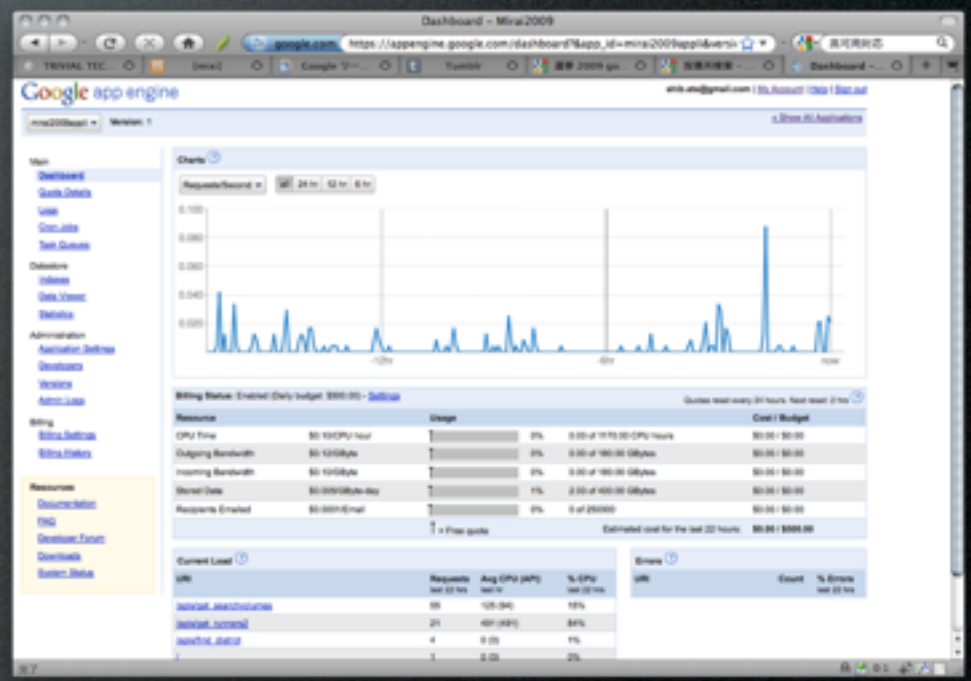
- App Engine自体がスケーラブル
  - Key Value型(KVS)データベース(非RDB)
  - 冗長化の仕組みを内蔵
- Googleが自社SaaSに利用する技術を使用
  - Google Apps, Gmail, YouTube etc.
- 負荷にあわせて自動的にスケールアウト

スケーラブルな設計

- DataStore(BogTable)の性質に合ったデータ設計
- アンチ正規化 - データの重複/肥大化を許す
  - Joinを避ける - 事前に計算しておけるデータはあらかじめ処理をしてデータとして展開する
- グローバルカウンタのようなロックを誘発するデータ構造を避ける

運用の手軽さ

- サーバのセットアップが不要
- OSなど低レイヤー部分の運用が不要
  - ただし監視は必要
- デプロイだけで修正がオンサイトに



チープ(1)

- App Engine課金の仕組み
  - データ転送量，CPU使用量，保存するデータのサイズ，メールの送信数で課金

リソース	単位	単価
発信帯域幅	GB	$0.12
受信帯域幅	GB	$0.10
CPU 時間	CPU 時間	$0.10
保存データ	GB/月	$0.15
メール受信者	受信者	$0.0001

チープ(2)

- App Engineの課金はかなり安い
  - 全立候補者の顔写真，データ
  - 400Gの投票所住所データ
  - かなりのアクセス - 膨大な配信量
- 課金は無料の範囲を少し超えたくらい

Pythonによる開発

- 開発効率の良い言語Python
  - 短いスケジュールで開発が可能
  - 仕様変更にも柔軟に対応

App Engineの
利点と欠点

利点1：信頼性

- Googleのソフトウエア，インフラを活用できる
  - 長年実働し，多くのユーザが使うGoogleのシステムとベースが同じ
- Googleがネット上で様々なサービスを大規模に展開しているインフラを活用できる

利点2:スケーラブルかつチープ

- スケーラブルなWebシステムを作るための費用対効果が高い
  - 開発コスト，運用コスト
- 多くのアクセスを集めるタイプのWebシステム
  - ソーシャルアプリ，ソーシャルゲーム

利点3:手軽さ

- 運用の手軽さ
  - サーバの構築，管理が不要
- 開発，デプロイが手軽
  - 開発環境でテスト，デプロイ
  - IDEを使った開発(Java)
  - スクリプト言語を使った効率の良い開発(Python)

欠点1:非RDBMS(1)

- リレーショナルデータベースの文脈が活用できない
  - ユニーク属性，インデックスのような機能が組み込みで存在しない
  - App Engineでは正規化はしばしば悪

欠点1:非RDBMS(2)

- DataStoreでの制限はスケーラビリティのトレードオフ
- RDBMSの高速化テクニックはバッドノウハウ
- DataStoreもRDBMSも，設計のエッセンスは根底では同じ

欠点2:システムを Googleに委ねる必要がある

- データ，コードがGoogle側に
  - 秘匿性の高いデータは扱いづらい
  - 内部統制
- アベイラビリティ，ダウンタイム
  - 現実的に問題にならないレベルだが、やはりサービスが止まることはある

欠点3：「箱庭」の窮屈さ

- 利用できるライブラリが制限される
- 外部への通信が制限される
- 超時間CPUを使うリクエストは途中でキャンセルされる
- etc.

まとめ

- 制限をかけた「箱庭」上で，スケーラブルでチープなサービスを構築できるのがApp Engine
- 利点をうまく生かして有効活用することが重要

ありがとうございました