Flex with Google App Engine for Java

Presented by wacky

FxUG in Toyama #1　2009.7.4

はじめに

- Flex楽しい～(2007年末から始めました)
- でも、本格的に作るならサーバが欲しいなぁ～
- サーバ持つとお金がかかるのよね…

- お？Javaが無料で使えるサーバが公開されている？
- さすがGoogleさん、太っ腹！
- さっそく試してみないと！

Google App Engine for Javaとは？

- クラウドコンピューティングプラットフォーム
  - 自動スケールアウト可能な、Webアプリケーション実行環境
  - ライバル
    - Amazon EC2(Elastic ComputeCloud)
    - Microsoft Windows Azure

- 2009/4/7にデビュー
  - Google App Engine２番目の開発言語
    - 初代はPythonで、2008年5月に提供されています
  - 現在Early Look
    - ２万５千人限定で公開中

- ほぼ無料？
  - 500万ページビュー/月程度に必要なリソースまでは無料
  - 2009/6/22に無料範囲が減りました…

価格表

リソース	無料(１日)	有料(１日)	従量課金
Requests	1,300,000	140,000,000	
- Outgoing	1 GB	(最大: 1046GB)	$0.12 / GB
- Incoming	1 GB	(最大: 1046GB)	$0.10 / GB
- CPU Time	6.5 CPU hours	(最大: 1729 hours)	$0.10 / CPU Hour
Stored Data	1 GB	(制限なし)	$0.15 / GB / Month
- Send	12 GB	72 GB	
- Received	115GB	695GB	
- CPU Time	60 CPU hours	1200 CPU hours	
E-mail	2,000 recipients	(最大:7,400,000)	$0.0001 / recipient
URL Fetch(API call)	657,000	46,000,000	
Deployments	250	250	

http://code.google.com/intl/ja/appengine/docs/quotas.htmlから抜粋＋α

Google App Engine for Java開発 (表)

- 一般的なJavaのInterfaceでプログラムが書ける
  - Java 6実行環境
  - Servlet / JSP
  - JDO 2.3, JPA 1.0　　BigTable
  - JavaMail
  - JCache(JSR 107)　　Memcache

- 開発は、Eclipse上で行う
  - ローカルでデバッグ実行可能
  - ボタンをクリック＆パスワード入力でデプロイ可能

Google App Engine for Java開発 (裏)

- ファイルシステムの実体はGFS(Google File System)
- データベースの実体はBigTable
  - Google提供サービスに裏打ちされたクラウド対応DB
  - スケーラビリティ命！
  - だから...
    - 最大1000件しか取得できないけど気にしないで
    - テーブルの結合検索苦手だけど気にしないで
      - テーブルの非正規化推奨！
    - 専用Indexが無いとソートできないけど気にしないで
      - 検索条件無し、ソート対象カラムが１つならソート可能...
      - Index追加しても直ぐに使えない仕様なのでよろしく～
    - Group byなんか無いけど気にしないで
    - グローバルトランザクション無いけど気にしないで
      - 同一グループに属するテーブル間のみトランザクション制御可

それって専用設計が
必要って事ですね...

その他の制限...

- リクエストは３０秒以内
  - Cron処理も３０秒以内...
  - Threadの作成禁止...
  - 実は５秒ぐらいでタイムアウトする場合あり...
- サーバにはファイル書き込み不可
  - 静的なファイルは問題なく配備できます
  - 読み込みは問題ありません
- JNIが使えない
- ネットワーク接続に制限あり

無料だから許す（何様

良い事も書いておこう...

- デプロイがすごく簡単
- 負荷に応じて自動的にスケールアウト
- ローカルデバッグ用の環境が提供されている
- 複数Versionを同時に実行可能
- アプリケーションの管理コンソールがある
  - リソース使用状況が詳細にわかる
  - URL別のアクセス回数、平均処理時間も分かる

ここからが本題です

AMFとは？

Action Message Formatの略です
Adobe LiveCycleDSをオープンソース化したBlazeDSが有名
（※最近、初めてAMF通信を使ったのは秘密です…）

- 長所
  - 自動でシリアライズ・デシリアライズ
    - カスタムクラスも自動変換
    - Flex側の処理がビルドインで高速
  - 通信データがコンパクト
    - gzip形式で圧縮してから送信
- 短所
  - Adobe独自形式
    - サーバ資産が、Flex専用になってしまう？
      - あまり依存しないみたいです（設定ファイルだけ依存？）
    - 仕様は公開されています
  - バイナリ形式で可読性が低い

BlazeDS (設定)

- 設定してみました…
- 動きません…(涙)

```
[amfsample/test.3345702802286194 42].<stderr>: ****
MessageBrokerServlet failed to initialize due to runtime exception:
Error: java.lang.NoClassDefFoundError:
java.lang.management.ManagementFactory is a restricted class. Please
see the Google App Engine developer's guide for more details.
```

Google App Engineが
サポートしていないJava APIを使用していますね…

The JRE Class White List
http://code.google.com/intl/ja/appengine/docs/java/jrewhitelist.html

ご清聴、
ありがとうございました

m(_ _)m

嘘です…ごめんなさい…

- もうちょっと調べてみました…
- GAEで実績のあるAMF通信可能なライブラリ
  - WebORB
    - http://www.themidnightcoders.com/blog/2009/04/running-weborb-for-java-in-google-app.html
    - http://www.themidnightcoders.com/products/weborb-for-java/overview.html
  - GraniteDS
    - http://graniteds.blogspot.com/2009/04/graniteds-20-on-google-app-engine.html
    - http://www.graniteds.org/confluence/display/INTRO/2009/06/18/Granite+Data+Services+2.0.0.GA+released
- でも、こんな記述もある…
  - http://martinzoldano.blogspot.com/2009/04/appengine-adobe-blazeds-fix.html

動きました！

- でも、制限が…
  - Remoting(RCP)しか動かないみたいです
  - サービスの同時実行が不安定

BlazeDS設定方法 (前提)

- お断り
  - 前述の制限がある設定方法です
- 前提
  - Eclipse 3.4 (Ganymede)
  - Adobe Flex Builder 3.0.2 Plug-in
  - Google App Engine for Java
    - ID登録(携帯必須)
    - 更新サイト: http://dl.google.com/eclipse/plugin/3.4
      - Google Plugin for Eclipse 3.4
      - Google App Engine SDK for Java 1.2.1 (2009/5/13)

BlazeDS設定方法 (本編)

- 設定手順
  - Web Application Project(GAE)を作成する
    - WEB-INF/appengine-web.xml
      - application
      - version
      - session-enabled
  - BlazeDSを追加
    - blazeds.warを、Webアプリケーションルートにインポート
    - WEB-INF/flex/services-config.xml
    - WEB-INF/flex/remoting-config.xml
  - Flex Projectを作成
    - http://learn.adobe.com/wiki/display/Flex/Creating+Flex+Builder+Projects+that+Use+Server+Technologies

BlazeDS設定方法 (暫定処置)

- ここまでの設定で、ローカル実行は可能です
  でも、GAEにデプロイして実行するとエラーに…
  下記２項目の暫定処置が必要です

- BlazeDSの管理機能を無効化
  - WEB-INF/flex/services-config.xml
    - manageableをfalseに変更
- セッション重複エラーを潰す
  - BaseHTTPEndpointを修正する

※BufferedInputStreamでラップする修正は不要

BlazeDS設定方法 (デモ)

- 下記のデモを予定
  - 実際に設定を行ったプロジェクトの参照
    - Hello Worldアプリ付き
  - Google App EngineへのDeployデモ
  - JDOを含む簡単なアプリケーションのデモ

http://appengine.google.com/ App Engineの管理コンソール

http://amfsample.appspot.com/ Sampleアプリケーション

実用的なサンプル(?)

- こんなのも提供しています…

http://mh.develop.jp/flex/

おわりに

- 自己紹介（今更
  - 富山で働いています
  - 社内の業務システムとか作っています
  - 開発言語はいろいろ
    - Java / Flex / PHP / C / C++ /...
- Googleさんに、お願い...
  - XMPP対応を早期リリースして下さい　m(_ _)m
- Google App Engine(BigTable)は非常識で楽しいので、是非、試してみてください～

Version (予備)

- Google App Engineのデプロイは...
  - アプリケーションID別 (最大１０個)
    - http://(application).appspot.com
  - Version別 (任意の名前)
    - http://(version).latest.(application).appspot.com

- Versionが違うと？
  - 複数のインスタンスを同時にサーバで動かせる
    - 本番
    - テスト中
    - メンテナンスツール(Pythonでも大丈夫)
      - インデックスの削除がPythonからしか行えなかったり...
  - Datastore, Memcacheは共有

BlazeDSで画像配信 (予備)

- 画像配信部分に関するTips

- 前提: 画像データはデータベースに保存
- 普通にBlazeDSで送ってみた
  - キャッシュされません...
    - ファイルに保存することは不可能...
    - メモリーに乗せておくには大きい...
  - 圧縮の恩恵も皆無だし...

- 結論: 素直にServletで送りましょう^^;
  - ブラウザのキャッシュが有効活用できます
  - URLを再利用しない設計がお勧めです

画像配信Servlet(サンプル)

```
public void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException {
    long since = request.getDateHeader("If-Modified-Since");
    String path = request.getPathInfo();
    String imageId = null;
    if (path != null && path.length() > 1) {
        imageId = path.substring(1);
    }

    PersistenceManager pm = PMF.getPersistenceManager();
    try {
        ImageData image = pm.getObjectById(ImageData.class, imageId);
        long createTime = image.getCreateTime().getTime();
        createTime -= createTime % 1000;
        if (since >= createTime) {
            response.setStatus(HttpServletResponse.SC_NOT_MODIFIED);
            return;
        }
        byte[] imageData = image.getImage().getBytes();
        response.setDateHeader("Last-Modified", createTime);
        response.setContentType("image/jpeg");
        response.setContentLength(imageData.length);
        response.getOutputStream().write(imageData);
    } catch (JDOObjectNotFoundException e) {
        response.sendError(HttpServletResponse.SC_NOT_FOUND);
    } finally {
        pm.close();
    }
}
```