Google App Engine: Cloud vs Cluster

Toby Reyelts
3/8/10

Goals

1. What
2. Why
3. How

What

A Cloud

- Not just run your software on our hardware
- All clouds are not the same
- Google App Engine is a platform, not just hardware

Why

- Simplicity
  - Development
  - Maintenance
- Price
  - Free to start
  - Only pay for what you use
- Scale on demand

The Platform

- Webapp based
  - Java Servlets (2.5)
  - JSP, JSTL, EL
  - HttpSessions
  - WAR deployment
- HTTP only
- 30 second request limit

appengine-web.xml

```
<appengine-web-app>
  <application>application-id</application>
  <version>1</version>
</appengine-web-app>
```

appengine-web.xml

```
<appengine-web-app>
  <application>application-id</application>
  <version>1</version>
  <static-files>
    <include path="/**.png" expiration="365d"/>
    <exclude path="/template/**.template"/>
  </static-files>
</appengine-web-app>
```

appengine-web.xml

```
<appengine-web-app>
  <application>application-id</application>
  <version>1</version>
  <static-files>
    <include path="/**.png" expiration="365d"/>
    <exclude path="/template/**.template"/>
  </static-files>
  <system-properties>
    <property name="max-length" value="140"/>
  </system-properties>
</appengine-web-app>
```

appengine-web.xml

```
<appengine-web-app>
  <application>application-id</application>
  <version>1</version>
  <static-files>
    <include path="/**.png" expiration="365d"/>
    <exclude path="/template/**.template"/>
  </static-files>
  <system-properties>
    <property name="max-length" value="140"/>
  </system-properties>
  <sessions-enabled>true</sessions-enabled>
</appengine-web-app>
```

Sandboxed

- Restrictions
  - JVM permissions
  - WhiteListed classes
- Why?
  - Clustering
  - Protect applications

Sandboxed

Restriction	Alternative
Threads	Async API Task Queue
Direct network connections	URLConnection
Direct file system writes	Memory, memcache, datastore
Java2D	Images API Software rendering
Native code	Pure Java Libraries

Sandboxed

Sandboxed

Developers need flexibility. JVM permissions are coarse.

Sandboxed

Developers need flexibility. JVM permissions are coarse.

They either provide a cramped sandbox.

Sandboxed

Developers need flexibility. JVM permissions are coarse.

They either provide a cramped sandbox.

Or they hand over the nuclear launch codes.

Sandboxed

App Engine delivers a happy medium.

Services

Service	Java Standard	Google Infrastructure
Authentication	Servlet API	Google Accounts
Datastore	JPA, JDO	Bigtable
Caching	JSR 107 javax.cache	memcacheg
Task Queues	N/A	Task execution server
Blobstore	N/A	Large object serving
E-mail	javax.mail	Gmail gateway
XMPP	N/A	Google Talk gateway
URLFetch	URLConnection	Caching HTTP proxy

Datastore

- Built-in
- Bigtable++. Not relational.
- Distributed, scalable, transactional, hierarchical
- APIs
  - Low Level
  - JDO and JPA subset

Datastore Entities

```
Entity car = new Entity("car");

car.setProperty("model","SL2");
car.setProperty("color", "Silver");
car.setProperty("mileage", 120452);
car.setProperty("drivers", Arrays.asList("Bracha",
"Gosling", "Gafter", "von der Ahé"));

Key carKey = datastoreService.put(car);
Entity copyOfCar = datastoreService.get(carKey);
```

Datastore Entities

```
Entity car = new Entity("car");

car.setProperty("model","SL2");
car.setProperty("color", "Silver");
car.setProperty("mileage", 120452);
car.setProperty("drivers", Arrays.asList("Bracha",
"Gosling", "Gafter", "von der Ahé));

Key carKey = datastoreService.put(car);
Entity copyOfCar = datastoreService.get(carKey);
```

Datastore Entities

```
Entity car = new Entity("car");

car.setProperty("model","SL2");
car.setProperty("color", "Silver");
car.setProperty("mileage", 120452);
car.setProperty("drivers", Arrays.asList("Bracha",
"Gosling", "Gafter", "von der Ahé"));

Key carKey = datastoreService.put(car);
Entity copyOfCar = datastoreService.get(carKey);
```

Datastore Queries

```
Query q = new Query("car");

q.addFilter("model", EQUAL, "SL2");
q.addFilter("mileage" LESS, 200000);
q.addFilter("drivers", "Bracha");
q.addSort("mileage", ASCENDING);

q.setKeysOnly();
PreparedQuery pq = dsService.prepareQuery(q);

for (Entity e : pq.asIterable()) {
  out.println(e.getKey());
}
```

Datastore Queries

```
Query q = new Query("car");

q.addFilter("model", EQUAL, "SL2");
q.addFilter("mileage" LESS, 200000);
q.addFilter("drivers", "Bracha");
q.addSort("mileage", ASCENDING);

q.setKeysOnly();
PreparedQuery pq = dsService.prepareQuery(q);

for (Entity e : pq.asIterable()) {
  out.println(e.getKey());
}
```

Datastore Queries

```
Query q = new Query("car");

q.addFilter("model", EQUAL, "SL2");
q.addFilter("mileage" LESS, 200000);
q.addFilter("drivers", "Bracha");
q.addSort("mileage", ASCENDING);

q.setKeysOnly();
PreparedQuery pq = dsService.prepareQuery(q);

for (Entity e : pq.asIterable()) {
  out.println(e.getKey());
}
```

Memcache

- Built-in
- Distributed, volatile, not transactional
- Store any Serializable object (up to 1M)
- No callbacks on eviction

Memcache

```
List getTweetsFor(String user) {
  String key = user + "-tweets";
  List tweets = (List) memcacheSvc.get(key);
  if (tweets == null) {
    tweets = getTweetsFromTwitter(user));
    memcacheSvc.put(key, tweets);
  }
  return tweets;
}
```

Memcache

```
// Prefer
memcacheService.putAll(map);

// Over
memcacheService.put(key1, item1);
memcacheService.put(key2, item2);
...
```

```
// Prefer
memcacheService.getAll(listOfKeys);

// Over
memcacheService.get(key1);
memcacheService.get(key2);
...
```

Blobstore

- Store big objects
- Receive and serve blobs asynchronously
- Work with small BlobKeys, not huge byte arrays

Blobstore

```
<body>
 <form
  action="<%=service.createUploadUrl("/upload")%>"
  method="post"
  enctype="multipart/form-data">
  <input type="file" name="myFile">
  <input type="submit" value="Submit">
 </form>
</body>
```

Blobstore

```
// Mapped to /upload in web.xml

public void doPost(
  HttpServletRequest req,
  HttpServletResponse res) {

  Map<String, BlobKey> blobs =
    blobstoreService.getUploadedBlobs(req);
  BlobKey blobKey = blobs.get("myFile");

  if (didntWantThisBlob) {
    blobstoreService.delete(blobKey);
  }
  ...
```

Blobstore

```
public void doGet(
  HttpServletRequest req,
  HttpServletResponse res) {

  BlobKey blobKey = new BlobKey("myFile");
  blobStoreService.serve(blobKey, res);
}
```

Task Queues

- Perform asynchronous processing
- Named queues
- Create tasks on demand
- Limited to 30s per task
- Can’t wait for a task.

Task Queues

```
// queue.xml
<queue-entries>
  <queue>
    <name>online</name>
    <rate>1/s</rate>
    <bucket-size>10</bucket-size>
  </queue>
  <queue>
    <name>bulk</name>
    <rate>2000/d</rate>
    <bucket-size>3</bucket-size>
  </queue>
</queue-entries>
```

Task Queues

```
// web.xml
<servlet>
  <servlet-name>report_servlet</servlet-name>
  <servlet-class>com.ReportServlet</servlet-class>
</servlet>

<servlet-mapping>
  <servlet-name>report_servlet</servlet-name>
  <url-pattern>/task/report_servlet</url-pattern/>
</servlet-mapping>

<security-constraint>
  <web-resource-collection>
    <url-pattern>/task/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <role-name>admin</role-name>
  </auth-constraint>
</security-constraint>
```

Task Queues

```
// Adding a new task
Queue queue = QueueFactory.getQueue(“bulk”);
queue.add(url("/task/report_servlet").
  param("type", “expense”).
  param(“user”, user).
  param(“id”, id));
```

```
// ReportServlet
void handleRequest(HttpServletRequest rq,
HttpServletResponse rp) {
  boolean finished = workOnReport(rq);
  if (!finished) {
    queueItAgain(rq);
  }
}
```

Demo

http://lotrepls.appspot.com

Getting Started

Getting Started

1. Download the SDK
   http://code.google.com/appengine

Getting Started

1. Download the SDK
   http://code.google.com/appengine

2. Play with sample on the dev_appserver

Getting Started

1. Download the SDK
   http://code.google.com/appengine

2. Play with sample on the dev_appserver

3. Sign up for an app_id
   http://appengine.google.com

Getting Started

1. Download the SDK
   http://code.google.com/appengine
2. Play with sample on the dev_appserver
3. Sign up for an app_id
   http://appengine.google.com
4. Upload the sample with your app_id using appcfg

Getting Started

1. Download the SDK
   http://code.google.com/appengine
2. Play with sample on the dev_appserver
3. Sign up for an app_id
   http://appengine.google.com
4. Upload the sample with your app_id using appcfg
5. Test at http://appid.appspot.com

Getting Started

1. Download the SDK
   http://code.google.com/appengine
2. Play with sample on the dev_appserver
3. Sign up for an app_id
   http://appengine.google.com
4. Upload the sample with your app_id using appcfg
5. Test at http://appid.appspot.com
6. Browse the admin console
   http://appengine.google.com

Getting Started

1. Download the SDK
   http://code.google.com/appengine
2. Play with sample on the dev_appserver
3. Sign up for an app_id
   http://appengine.google.com
4. Upload the sample with your app_id using appcfg
5. Test at http://appid.appspot.com
6. Browse the admin console
   http://appengine.google.com
7. Profit

Questions

Attributions

- Creative Commons 2.0
- Right-Sized Sandbox
  http://www.flickr.com/photos/23258385@N04/2237739552
- Cramped Sandbox
  http://www.flickr.com/photos/lizhenry/2591584958
- Nuclear Bomb

  http://www.flickr.com/photos/blatantnews/3950701630