Google AppEngine

Mastering Cloud Computing
Chapter 9.2
Paul Talaga

UNIVERSITY OF Cincinnati

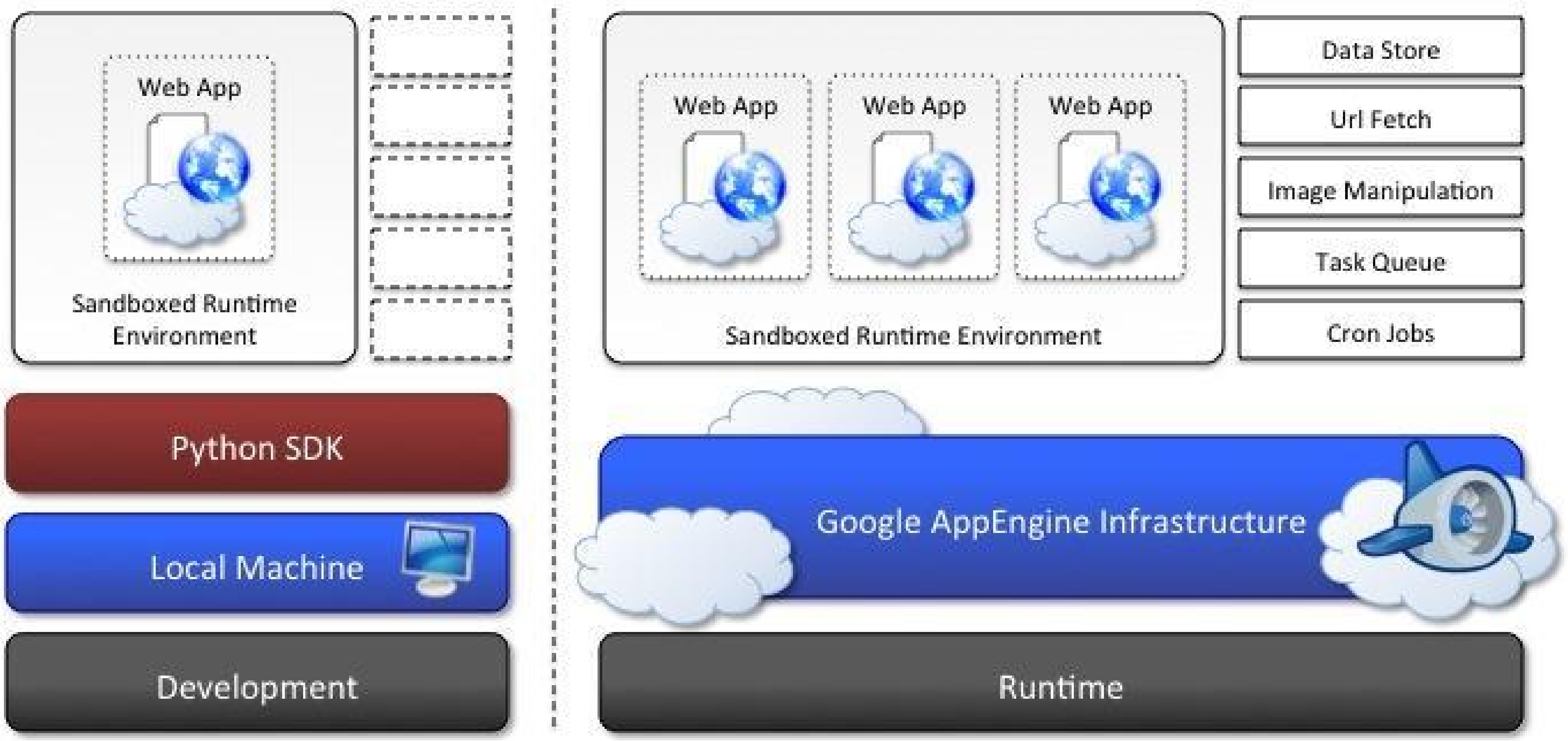
AppEngine Facts/History

- PaaS Service! (They provide hardware)
- Preview in 2008, full in 2011
- UP in abstraction from IaaS.. MUST use their languages and versions!
  - Python, Java, Go, and PHP
- No ability to tune hardware
- But... Application autoscaling!
- Requires:
  - Stateless application
  - Use their storage system (NoSQL/SQL)

*Experimental

UNIVERSITY OF Cincinnati

Architecture



Limitations / Costs

- Vendor Lock In - No mature way to run such an app outside Google
- Limited languages
- Can’t drop-in old applications
- 60 sec/request max
- HTTP response size: 32MB
- Datastore Item: 1MB
- Costs:
  - Free: 28hrs/month, 100 emails, unlimited in, 1GB out, 1GB datastore, [...]

UNIVERSITY OF Cincinnati

Costs (cont)

- Extra features/performance from paid:
  - $0.05 - $0.30 / hr, billed to the minute
  - $0.12 / GB Data out to internet, in free
  - $0.06 per 100k datastore queries
  - $0.18 per GB / month Search storage
  - Can set daily $ limits - queries will fail after

AppEngine SDK on Ubuntu

- Python 2.7 is required!
  If `python` does not say 2.7.x, install Python 2.7.
- Now download Google AppEngine SDK
  https://cloud.google.com/appengine/downloads
- For convenience…..
  - Add `export PATH=$PATH:<the path>/google_appengine` in `.bashrc`
  - Close your terminal window and load a new. `dev_appserver.py` should now work.
  - Now you can do `$ dev_appserver.py <folder name>` from anywhere.

Demo! (sdk)

- Very simple Python webapp
- Many frameworks available: Django, Flask, etc … I'll use straight Python
- Running on development machine
  - `dev_appserver.py <folder>`
- Running on Google Hardware
  - Get Google Account, agree to terms
  - Create new app: `appengine.google.com`
  - Create/note application identifier!
  - Place identifier in `app.yaml` file (from template)
  - `appcfg.py -A <identifier> update <folder>`

Go to `<identifier>.appspot.com` !!!!

AppEngine Instances

- Like EC2 VM, but single process - 1 request at a time (unless threaded)
- Automatically will spawn new based on response time
- The better your app (fast response) fewer instances -> less costs!

Storage (built-in)(docs)

- Static files - .css, images, html, etc…
  - specify static files/folders in app.yaml
- Cloud (My)SQL - No free tier (cost)
- AppEngine Data Store (schemaless)
- Memcache
- Google Cloud Storage (files)

UNIVERSITY OF Cincinnati

Application Services (docs)

- UrlFetch - Web application can make requests
  - Can set deadline (5s default, 60s max)
  - Synchronous or asynchronous
  - Recursion not allowed
  - Max 10MB out, 32MB in
  - Uses Google's network!
- Memcache
- Image Manipulation - Requires PIL for local manip
- Mail & Instant Messaging
- User Management - `.nickname(), .email(), .user_id()`
- Channel - persistent connections (prevent polling)

UNIVERSITY OF Cincinnati

Using Memcache (docs)

- In-memory distributed key/value store
- Typical workflow
  a. App/fn get query
  b. Use memcache to see if query result is stored
     - If so, return that
     - If not, perform query
     - Store query result in memcache for later
- Can be used for increment/decrement
- Values must be `pickle`-able

```
from google.appengine.api import memcache
memcache.get(<key>)             <key> must be a string!
memcache.set(<key>, <value>)   Optional time in seconds.
memcache.add(<key>, <value>)
memcache.incr(<key>)            Optional value
```

UNIVERSITY OF Cincinnati