Clojure on Google App Engine

First steps

Chris Wilson
chris@cjetech.co.uk
@minleychris

Rodrigo Pimentel
rbp@isnomore.net
@rbp

What is App Engine

- Paas Cloud computing
- JVM, Python, Go
- Sandboxed
- Autoscaling
- Services and APIs
- Free(ish)

Services and APIs

- Blobstore
- Capabilities
- Channel
- **Datastore**
- Datastore Async
- Images
- **Mail**
- **Memcache**
- Multitenancy
- Remote
- SSL access
- Task Queue
- URLFetch
- **Users**
- XMPP
- etc, etc, etc...

Limitations

- No filesystem access
  - read-only access to *war* resources
- The JRE Class White List
  - Not whiteslisted, for instance:
    - Threads
    - Sockets
    - Files

Alternatives

- Amazon Elastic Beanstalk
- Heroku
- Redhat OpenShift
- etc...

Minimal requirements

- Web app using Ring
- WEB-INF/appengine-web.xml
- Make war directory (unzip the uberwar)
- App Engine Development Kit to upload

WEB-INF/appengine-web.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<appengine-web-app xmlns="http://appengine.google.com/ns/1.0">

  <application>crowdsort</application>
  <version>1</version>
  <static-files/>
  <resource-files/>
  <inbound-services>
    <service>mail</service>
  </inbound-services>

  <threadsafe>true</threadsafe>

</appengine-web-app>
```

appengine-magic

https://github.com/gcv/appengine-magic

- Management of xml and preparation
- Clojure wrappers for services
- Repl serving

appengine-magic

```
(ns example.core
  (:use compojure.core)
  (:require [appengine-magic.core :as ae]))

(defroutes example-app-handler
  (GET "/" req
       {:status 200
        :headers {"Content-Type" "text/plain"}
        :body "Hello, world!"}))

(ae/def-appengine-app example-app #'example-app-
handler)
```

appengine-magic - setup

```
lein new
```

dev-dep: `[appengine-magic "0.5.0"]`

```
lein deps
lein appengine new
```

appengine-magic - repl

```
(require '[appengine-magic.core :as ae])

(ae/serve foo-app)

(ae/start foo-app)
(ae/stop)
(ae/start foo-app :port 8095)
(ae/stop)
```

crowdsort - algorithm

- Crowdsource your sorts
- $O(\infty)$, $\Omega(1)$
- Keep or swap
- When is it sorted?
  - Crowdsource it

crowdsort - algorithm

- Crowdsource your sorts
- O(∞), Ω(1)
- Keep or swap
- When is it sorted?
  - Crowdsource it
- Patent pending

```
(defroutes crowdsort-app-handler
  (GET "/" req
       {:status 200
        :headers {"Content-Type" "text/html"}
        :body (swap-or-not-page)})
  (POST "/" {params :params} (handle-post params)))

  (route/resources "/")
  (route/not-found
       {:status 404
        :headers {"Content-Type" "text/plain"}
        :body "not found"}))

(ae/def-appengine-app crowdsort-app
                      (wrap-params crowdsort-app-handler))
```

```
(defn swap-or-not-page []
  (let [id (get-identifier)
        [{i1 :index v1 :value} {i2 :index v2 :value}]
                               (get-items-to-swap id)]  ; And lock!
    (html
     [:div (str v1 " " v2)]
     [:form {:method "POST"}
      [:input {:name "identifier" :value id :type "hidden"}]
      [:input {:name "index1" :value i1 :type "hidden"}]
      [:input {:name "index2" :value i2 :type "hidden"}]
      [:input {:name "action" :value "swap" :type "submit"}]
      [:input {:name "action" :value "keep" :type "submit"}]]
     [:div (str "Current list:" (seq (get-list)))])))
```

- Run it locally!
  - *lein appengine-prepare && dev_appserver.sh war*

```
(defn handle-post [{action "action"
                    index1 "index1" index2 "index2"
                    id "identifier"}]
  (process-action (Integer/parseInt index1)
                  (Integer/parseInt index2)
                  (Integer/parseInt id)
                  action)
  (swap-or-not-page))


(defn process-action [i j id action]
  (if (is-lock-held i j id)
    (do (if (swap? action)
          (swap-values i j)
          (do (keep-values i j)
              (process-sorted-list)))
        (unlock-indexes i j))))
```

"But what about App Engine?"

```
(ns crowdsort.core
  (:require [appengine-magic.core :as ae]
            [appengine-magic.services.memcache :as memcache]))

(defn get-list []
  (if (not (memcache/contains? "current-list"))
    (get-new-list))
  (memcache/get "current-list"))

(defn lock-index [idx id]
  (memcache/put! (get-lock-key idx) id
      :expiration (. com.google.appengine.api.memcache.Expiration
byDeltaSeconds 30)))

(defn unlock-index [idx]
  (memcache/delete! (get-lock-key idx)))
```

```
(defn process-sorted-list []
  (if (list-sorted?)
    (do (invalidate-all-locks)
     (reset-keep-counter)
     (notify-list-owner)
     (get-new-list))))

(defn number-of-keeps []
  (memcache/put! "keeps" 0 :policy :add-if-not-present)
  (memcache/get "keeps"))

(defn list-sorted? []
  (>= (number-of-keeps)
      (count (get-list))))
```

- It's easy to prove the stop condition above. All you need to do is make

"But where does the list come from?"

```
(defroutes crowdsort-app-handler
  (GET "/submit" req
       {:status 200
        :headers {"Content-Type" "text/html"}
        :body (submit-new-list-page)})
  (POST "/submit" {params :params}
        (handle-submit params))

(defn swap-or-not-page []
  [:div
    (if (user/user-logged-in?)
      (do [:div (link-to "/submit" "Submit an array")]
            [:div (link-to (user/logout-url) "Logout")]])
      (link-to (user/login-url) "Login to submit an array!"))]]
```

```
(defn submit-new-list-page []
  (html5
    [:div "Split items with one or more spaces:"
      [:form {:method "POST"}
      [:input {:name "array" :type "text"}]
      [:input.btn.btn-primary {:type "submit"}]]]))

(defn handle-submit [{array "array"}]
  (if (user/user-logged-in?)
    (let [new-list (split array #"\s+")]
       (let [list-to-sort (ListToSort. new-list
                                       (.getEmail (user/current-user))
                                       (.getTime (java.util.Date.)))]
         (ds/save! list-to-sort))))
  (redirect "/"))
```

```
(:require [appengine-magic.services.datastore :as ds])

(ds/defentity ListToSort [elements owner-email
                          submission-time])

(defn get-new-list []
  (let [new-list (first (ds/query :kind ListToSort
                                  :sort [:submission-time]))]
    (memcache/put! "current-list" (:elements new-list))))
```

- See for yourself!
  - App Engine's admin console.

"But how will I know when my list is crowdsorted?"

```
(:require [appengine-magic.services.mail :as mail])

(defn process-sorted-list []
    (notify-list-owner))

(defn notify-list-owner []
  (let [email-addr (get-list-owner-email)
        msg (mail/make-message
                :from "crowdsort@crowdsort.appspotmail.com"
                :to email-addr
                :subject "Your list has been crowdsorted!"
                :text-body (get-list))]
      (mail/send msg)))
```

"But..."

No more "but"s! It's ready!

- Run it locally!
  - *lein appengine-prepare && appcfg.sh update war*

Crowdsort

http://crowdsort.appspot.com/

https://github.com/chrisjwilson/crowdsort