JBoss Community

immutant

Jim Crossley
jcrossley@redhat.com

JBoss Community

Agenda

- Who, what, WHY?
- A fake, real-life application
- How?
- API deep-dive
- Clustering, etc

Who?

- Jim Crossley - @jcrossley3
- Toby Crawley - @tcrawley

project:odd

- Bob McWhirter - Director of Polyglot
- JBoss by Red Hat

What?

- An app server for Clojure
- Inspired by TorqueBox
- Powered by JBoss AS7
- Started in September 2011

App Server?

- Multiple apps running in same JVM
- Isolation via classloaders
- Shared common services

Inspired by TorqueBox

- Make Ring, not War
- Clojure interfaces to JBoss middleware
- No XML
- Adapt AS7 to the Clojure development common practices

AS7 is teh awesome!

- Tomcat for web
- HornetQ for messaging
- Infinispan for caching
- Quartz for scheduling
- Distributed Transactions (XA)
- Clustering
- Management/Monitoring

Certified, fwiw

Java EE 6 Full Profile

JBoss Community

Why?

- Reduce incidental complexity
- Simplify deployment of non-trivial applications
- Simplify clustering to achieve high-availability
- Encapsulate JBoss services
- Exploit the JVM

Polyglot

- Leverage the best tool on a single platform, regardless of language
- “ruby in the front, java in the middle, clojure in the rear”
- Red Hat is committed to it

Maybe a fit?

- If you're already using Jetty, Memcached, RabbitMQ, and/or Cron
- If you have a polyglot organization, especially Ruby and Java
- If you must integrate with legacy Java EE

Let’s Get Real

CRUD can only take you so far

JBoss Community

Apps often require...

- More than one web interface, e.g. administration
- Background processing
- Periodic maintenance
- Dynamic data sources, e.g. REST
- Services

But...

We'd still like to treat the integration of those components as a single, versioned application.

Consider...

- A tweet-analysis app
- Filters the Twitter streaming API for brand keywords
- Scrapes URL's from the tweets
- Stores weighted results of sentiment analysis

Possible Components...

- A web app for displaying impressions
- An admin app for defining users
- The Twitter streaming client
- The URL scraper
- A housekeeper to purge old data

The Web Apps

```
;; Mount the ring handlers
(web/start “/” brandy.web/app)
(web/start “/admin” brandy.web/admin)
```

JBoss Community

The Message Queue

```
;; Where scrapers pick up their work
(msg/start “/queue/tweets”)
```

```
;; The scrapers
(msg/listen “/queue/tweets”
            brandy.msg/scrape)
```

The Twitter Service

```
;; The work producer
(daemon/start “streamer”
              brandy.stream/start
              brandy.stream/stop)
```

JBoss Community

The Recurring Job

```
;; The housekeeper
(job/schedule “housekeeper”
              “0 0 0 * * ? ”
              brandy.job/purge)
```

immutant.clj

```
; The ring handlers
(web/start “/”      brandy.web/app)
(web/start “/admin” brandy.web/admin)

; Where scrapers pick up their work
(msg/start “/queue/tweets”)

; The scrapers
(msg/listen “/queue/tweets” brandy.msg/scrape)

; The work producer
(daemon/start “streamer” brandy.stream/start
                         brandy.stream/stop)
; The housekeeper
(job/schedule “housekeeper” “0 0 0 * * ?”
                            brandy.job/purge)
```

immutant.clj

```
; The ring handlers
(web/start "/"       brandy.web/app)
(web/start "/admin" brandy.web/admin)

; Where scrapers pick up their work
(msg/start "/queue/tweets")

; The scrapers
(msg/listen "/queue/tweets" brandy.msg/scrape)

; The work producer
(daemon/start "streamer" brandy.stream/start
                         brandy.stream/stop)
; The housekeeper
(job/schedule "housekeeper" "0 0 0 * * ?"
                            brandy.job/purge)
```

JBoss Community

immutant.clj

```
; The ring handlers
(web/start "/"      brandy.web/app)
(web/start "/admin" brandy.web/admin)

; Where scrapers pick up their work
(msg/start "/queue/tweets")

; The scrapers
(msg/listen "/queue/tweets" brandy.msg/scrape)

; The work producer
(daemon/start "streamer" brandy.stream/start
                         brandy.stream/stop)
; The housekeeper
(job/schedule "housekeeper" "0 0 0 * * ?"
                            brandy.job/purge)
```

Immutant

how?

JBoss Community

Leiningen Plugin

```
$ lein plugin install lein-immutant 0.6.0
```

JBoss Community

Install

```
$ lein immutant install [version]
```

Install

By default, the plugin installs Immutant beneath

```
~/.lein/immutant/
```

Run

```
$ lein immutant run
```

Deploy

```
$ lein new myapp
$ cd myapp

$ lein immutant init
$ $EDITOR immutant.clj

$ lein immutant deploy
```

Deploy

A “deployment descriptor” is a glorified symlink written to `jboss/standalone/deployments`

Its contents:

```
{:root "/the/path/to/your/app"}
```

Deploy

Each app gets a dedicated Clojure runtime and classpath containing resolved dependencies, src, lib, and resources.

project.clj

```
(defproject myapp "1.2.3"
  ...
  :immutant {:init myapp.core/initialize
             :resolve-dependencies true
             :context-path "/"
             :swank-port 4111
             :nrepl-port 4112})
```

Overlay

```
$ lein immutant overlay torquebox

$ export TORQUEBOX_HOME=$IMMUTANT_HOME
```

The API

immutant namespaces

JBoss Community

The API

- Dynamic by design
  - No static xml files or annotations
- Modules
  - Web
  - Messaging
  - Caching
  - Scheduling
  - Daemons

immutant.web

ring

JBoss Community

Web

- Supports any Ring-based handler
- Ring middleware to expose the Servlet session
- Multiple web apps may share the same lifecycle via “subcontexts”

Web

```
(require '[immutant.web :as web])

(defn my-handler [request]
  {:status 200
   :headers {"Content-Type" "text/html"}
   :body "Hello from Immutant!"})

(web/start "/hi" my-handler)
```

Context paths

project.clj

```
(defproject myapp "1.2.3" ...)
```

immutant.clj

```
(web/start "/" main)
;; http://localhost:8080/myapp/

(web/start "/admin" admin)
;; http://localhost:8080/myapp/admin
```

Context paths

project.clj

```
(defproject myapp "1.2.3”
  :immutant {:context-path "/"})
```

immutant.clj

```
(web/start "/" main)
;; http://localhost:8080/

(web/start "/admin" admin)
;; http://localhost:8080/admin
```

Session Replication

```
(require ‘[immutant.web :as web]
         ‘[immutant.web.session :as iws])

(web/start "/"
  (ring-session/wrap-session
    handler
    {:store (iws/servlet-store)}))
```

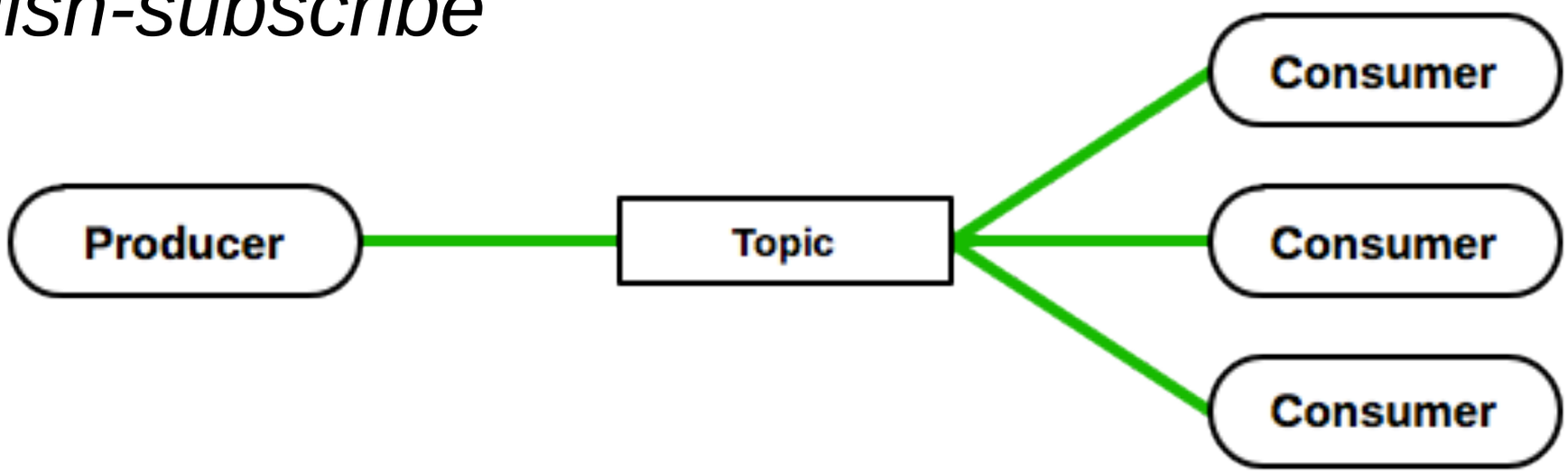
immutant.messaging

hornetq

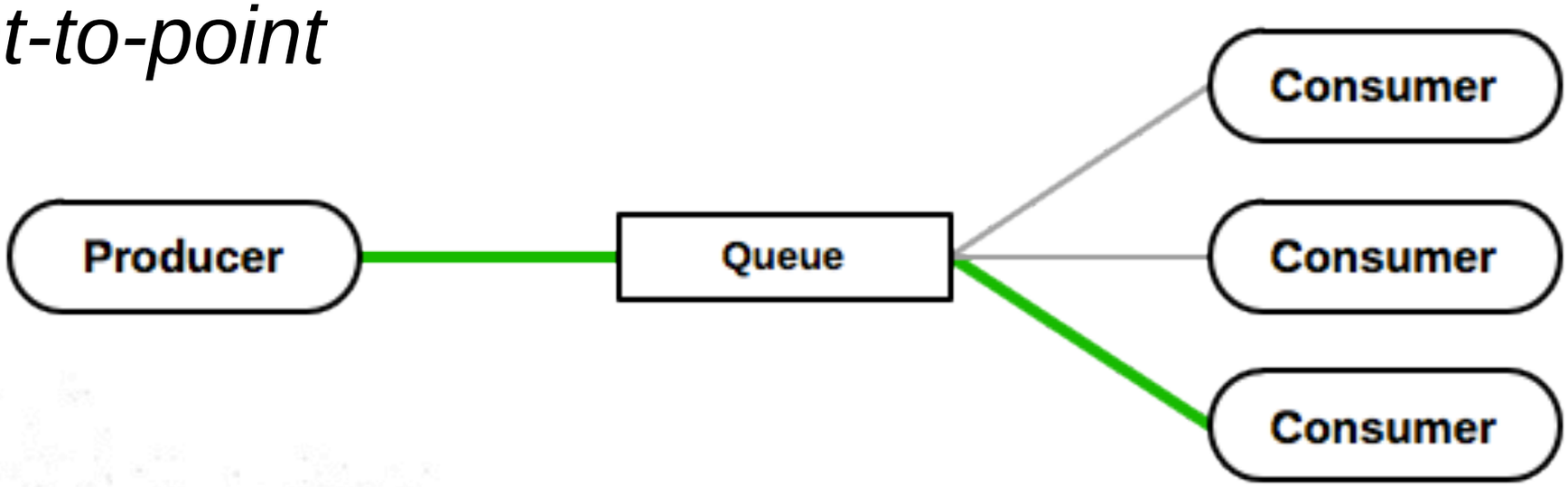
JBoss Community

Messaging

publish-subscribe



point-to-point



Messaging

```
(require '[immutant.messaging :as msg])

(msg/start "/queue/work")
(msg/start "/topic/news")
```

JBoss Community

Producers

```
;; Content may be any serializable
;; data structure

(msg/publish “/queue/work” anything)
```

Producers

```
;; May be as complex as necessary

(msg/publish "/queue/work"
  {:a "b" :c [1 2 3 {:foo 42}]})
```

Producers

```
;; Support priority and time-to-live

(msg/publish “/queue/work”
             a-message
             :priority :high
             :ttl 1000)
```

Producers

```
;; Three message encodings:
;; :clojure :json :text
```

```
(msg/publish "/queue/work"
  {:a "b" :c [1 2 3 {:foo 42}]}
  :encoding :json)
```

Messaging

With TorqueBox, messages comprised of standard collections and types are transparently interoperable between Clojure and Ruby in either direction.

Consumers

```
;; Block until message arrives
(let [task (msg/receive "/queue/work")]
  (perform task))
```

JBoss Community

Consumers

```
;; Create a lazy sequence of messages
(take 4 (msg/message-seq “/queue/foo”))
```

Consumers

```
(defn upper-caser [s]
  (msg/publish "/topic/upper"
               (.toUpperCase s)))

;; Register a listener
(msg/listen "/queue/lower" upper-caser)
```

Messaging

Automatic retries, failover and load-balancing when clustered.

JBoss Community

immutant.cache

infinispan

JBoss Community

Distributed Cache

- Backed by Infinispan data grid
- Implements core Clojure interfaces
  - `core.cache/CacheProtocol`
  - `core.memoize` store

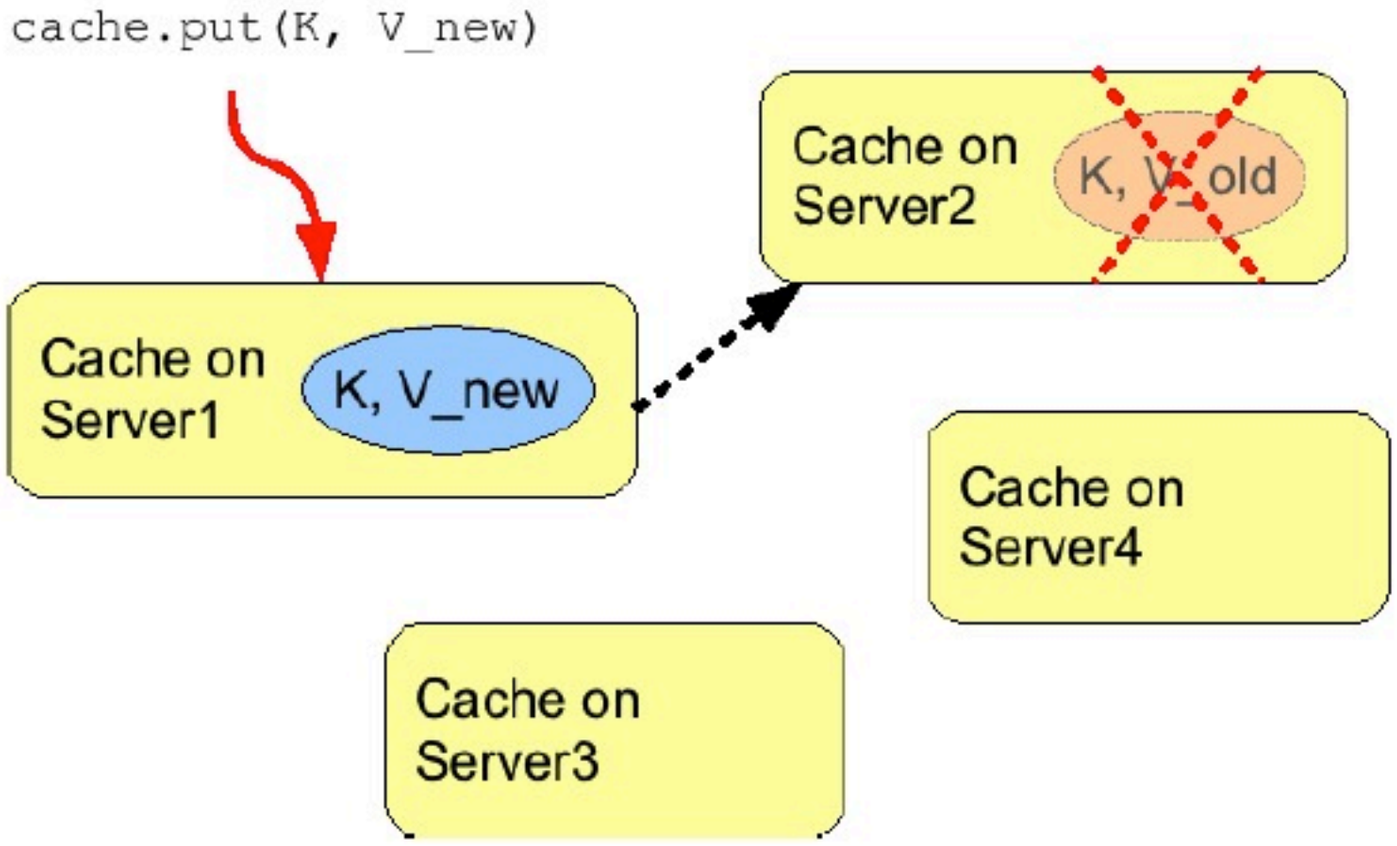
Infinispan

- Eviction: FIFO, LRU
- Expiration: time-to-live, idle time
- Persistence: write-through/behind
- Transactional: JTA/XA
- MVCC-based concurrency
- Flexible clustering

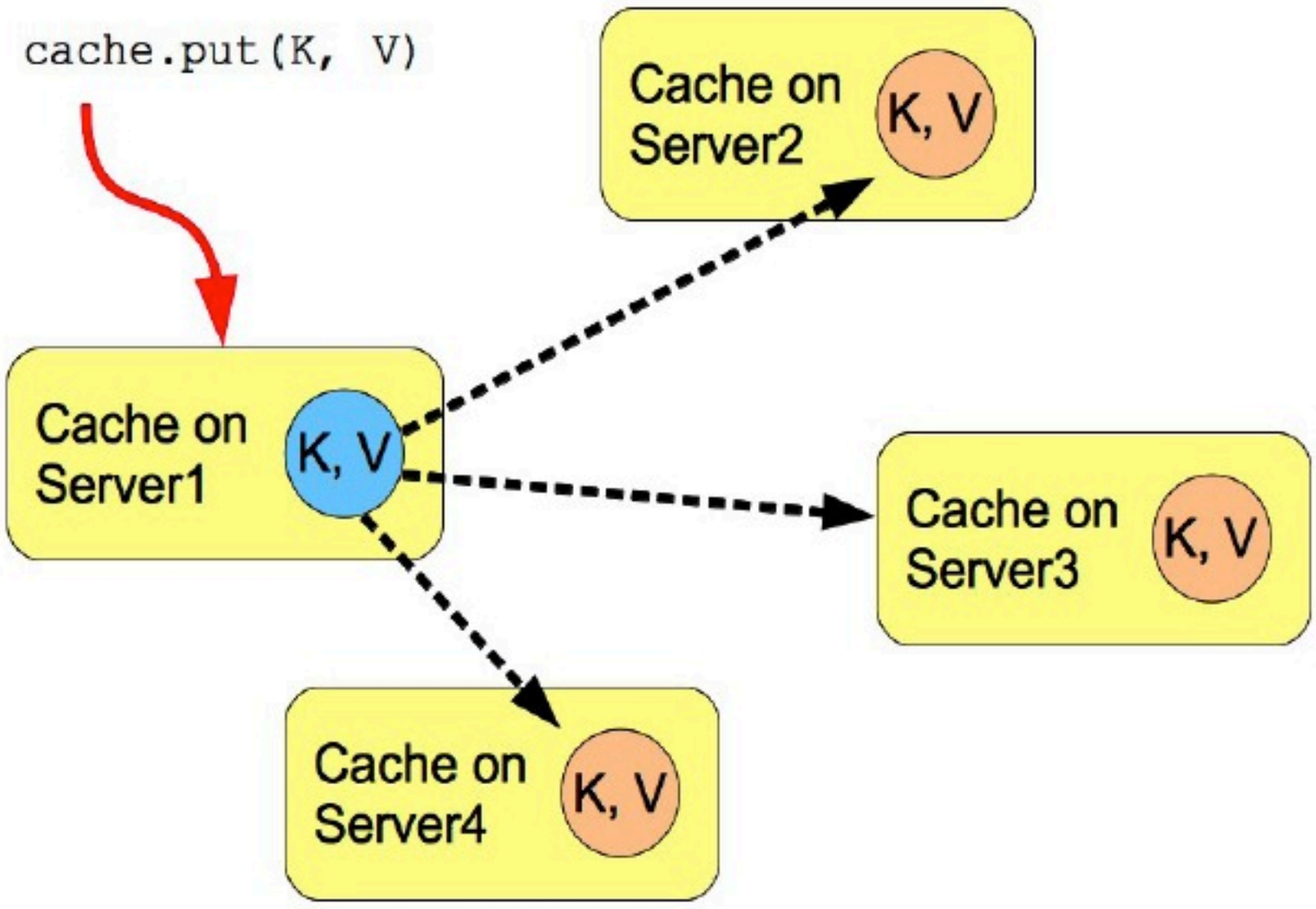
Cache Modes

- `:local` - non-clustered
- Clustered
  - `:invalidated` - no data shared
  - `:replicated` - all data shared
  - `:distributed` - linear scalability

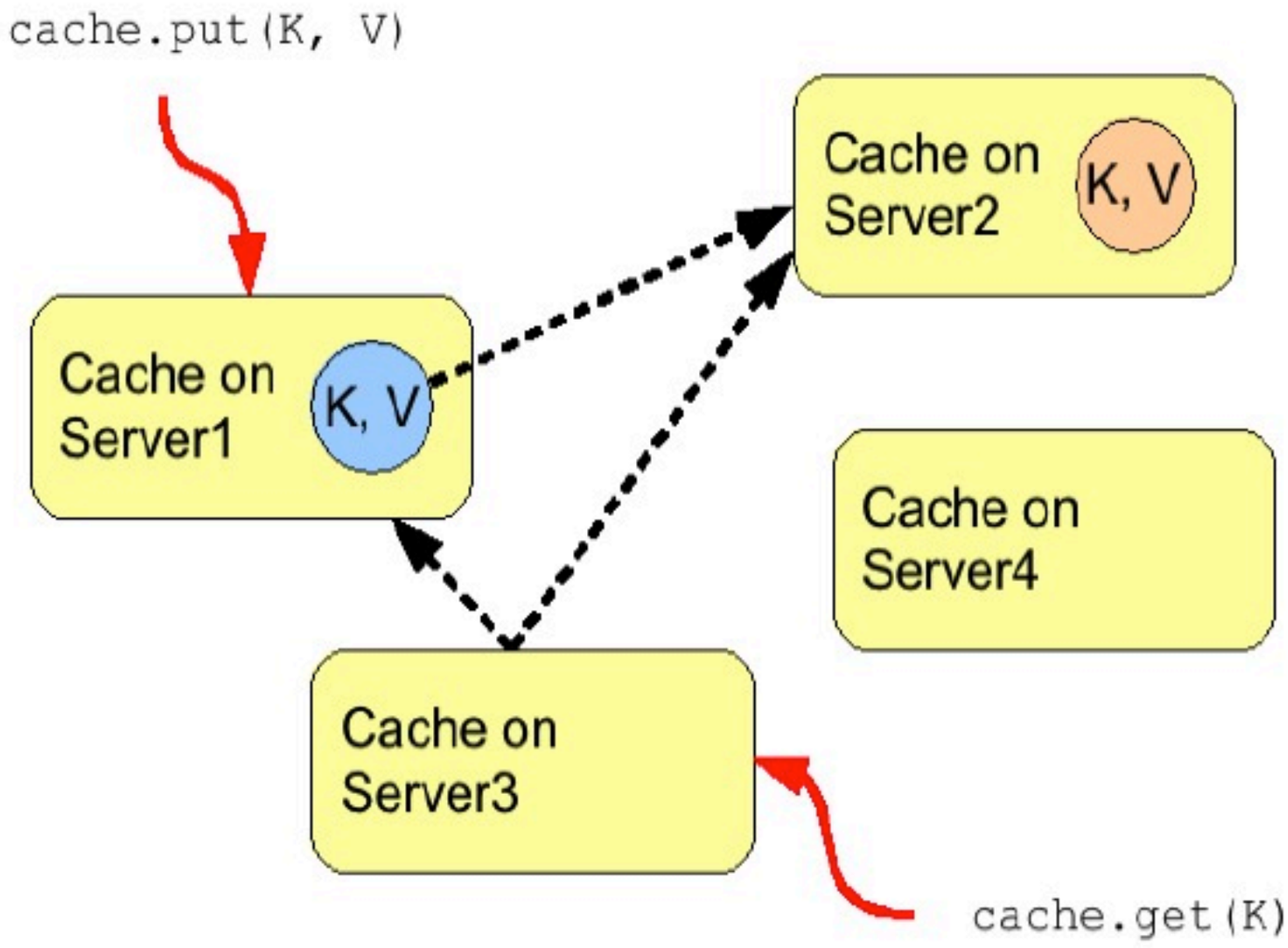
:invalidated



:replicated


cache.put(K, V)
Cache on Server1
K, V
Cache on Server2
K, V
Cache on Server3
K, V
Cache on Server4
K, V

:distributed



immutant.cache

- All caches have a name and a mode (:local, :invalidated, :replicated, :distributed)
- Like-named Immutant caches are backed by the same Infinispan cache

immutant.cache

```
(require '[immutant.cache :as ic])

(def a (ic/cache "foo" :replicated))
(def b (ic/cache "bar"))
(def c (ic/cache "foo"))
```

immutant.cache

```
(def c (ic/cache “foo”))
(assoc c :a 1)
(merge c {:b 2 :c 3})
(dissoc c :c)

c => {:a 1, :b 2}
```

Memoization

```
(require ‘[immutant.cache :as ic])

(def memoized-fn (ic/memo slow-fn “foo”))
(memoized-fn 1 2 3)
(def c (ic/cache “foo”))

c => {[1 2 3] 42}
```

Explicit writes

```
(require ‘[immutant.cache :as ic])

(def opts {:ttl 2 :idle 1 :units :days})
(def c (ic/cache “foo”))
```

```
(ic/put c key value opts)
(ic/put-if-absent c key value opts)
(ic/put-if-present c key value opts)
(ic/put-if-replace c key old new opts)
(ic/delete c key)
```

immutant.jobs

quartz

Scheduling

```
(require '[immutant.jobs :as job])

;; Once a month
(job/schedule "newsletter"
              "0 0 0 1 * ?"
              news/monthly
              :singleton true)
```

JBoss Community

Scheduling

"Fire every half hour from 10am until 1pm on the third Friday of each month"

0 */30 10-13 ? * FRI#3

Seconds	Minutes	Hours	DOM	Month	DOW	Year
0-59	0-59	0-23	1-31 ? L W	1-12 JAN-DEC	1-7 SUN-SAT ? L #	1970-2099 empty

Scheduling

```
(require '[immutant.jobs :as job])

(job/at "name" "3m"
  #(println "I fire after 3 minutes"))

(job/every "name" "1d"
  #(println "I fire once a day"))
```

immutant.daemons

long-running services

JBoss Community

Daemons

```
(require '[immutant.daemons :as daemon])

(daemon/start "name" start-fn stop-fn)
```

Daemons

```
(def response (atom nil))

(defn start []
  (reset! response
    (twitter-statuses-filter
      (terms)
      url-scraper)))

(defn stop []
  ((:cancel (meta @response))))

(daemon/start "tweets" start stop
              :singleton true)
```

Clustering

simple horizontal scaling

JBoss Community

Clustering

```
$ lein immutant run --clustered
```

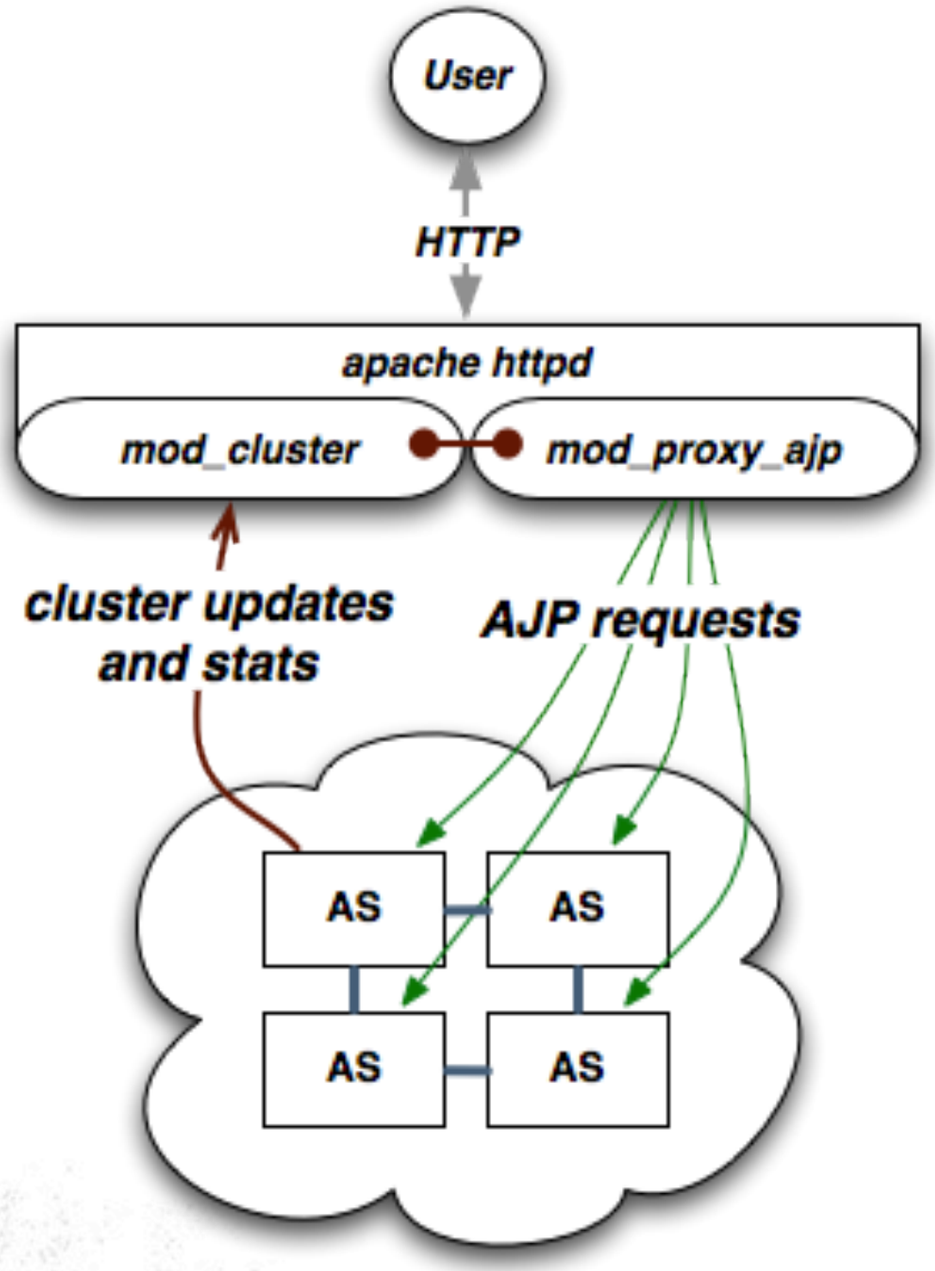
Clustering

- Session replication
- Robust, load-balanced message distribution
- HA Singleton
- HA Jobs
- Infinispan

mod_cluster

- An Apache module aware of JBoss
- Reverse proxy via AJP or HTTP[S]
- Uses load statistics and deployment availability for intelligent routing
- Optional session affinity
- Failover

mod_cluster


User
HTTP
apache httpd
mod_cluster
mod_proxy_ajp
cluster updates
and stats
AJP requests
AS
AS
AS
AS

Coming soon

- Distributed Transactions (XA)
- OpenShift integration
- Websockets
- Better remote support
- Domain Mode (cluster management)

http://immutant.org

#immutant

@immutants

JBoss Community