The Other mod_rails: Easy Rails Deployment with JRuby

Nick Sieger
Sun Microsystems, Inc

MATT GROENING

CGI/FCGI

Mongrel

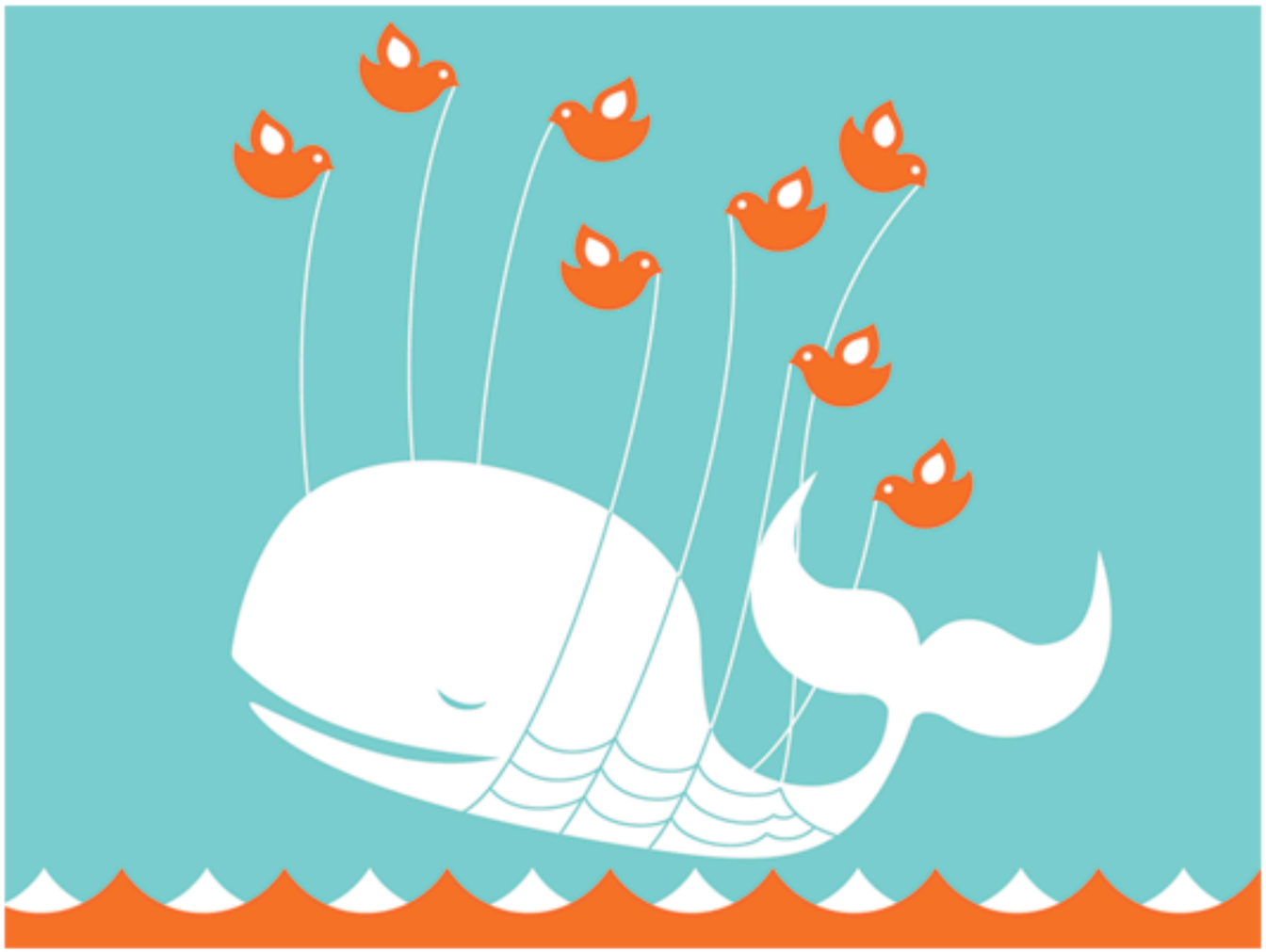
omg puppiez!

twitter

Home Public Timeline Help

Twitter is over capacity.

Too many tweets! Please wait a moment and try again.



not nirvana...

GOD

(like monit, only awesome)

A BETTER WAY TO MONITOR

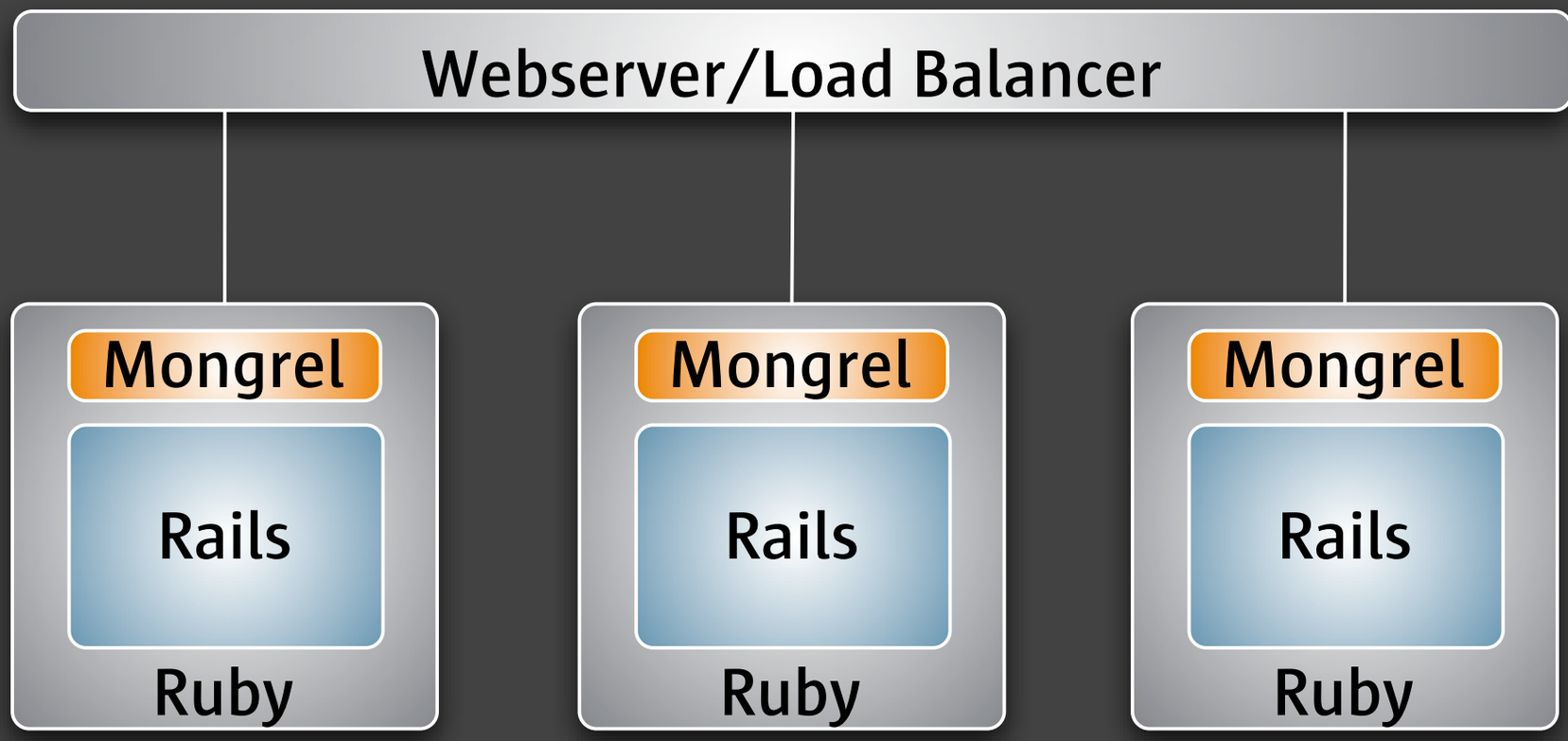
God is an easy to configure, easy to extend monitoring framework written in Ruby.

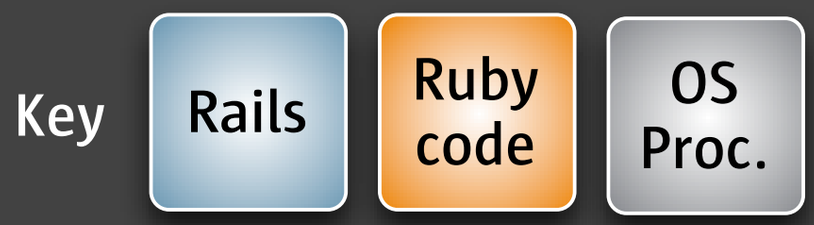
FEATURES

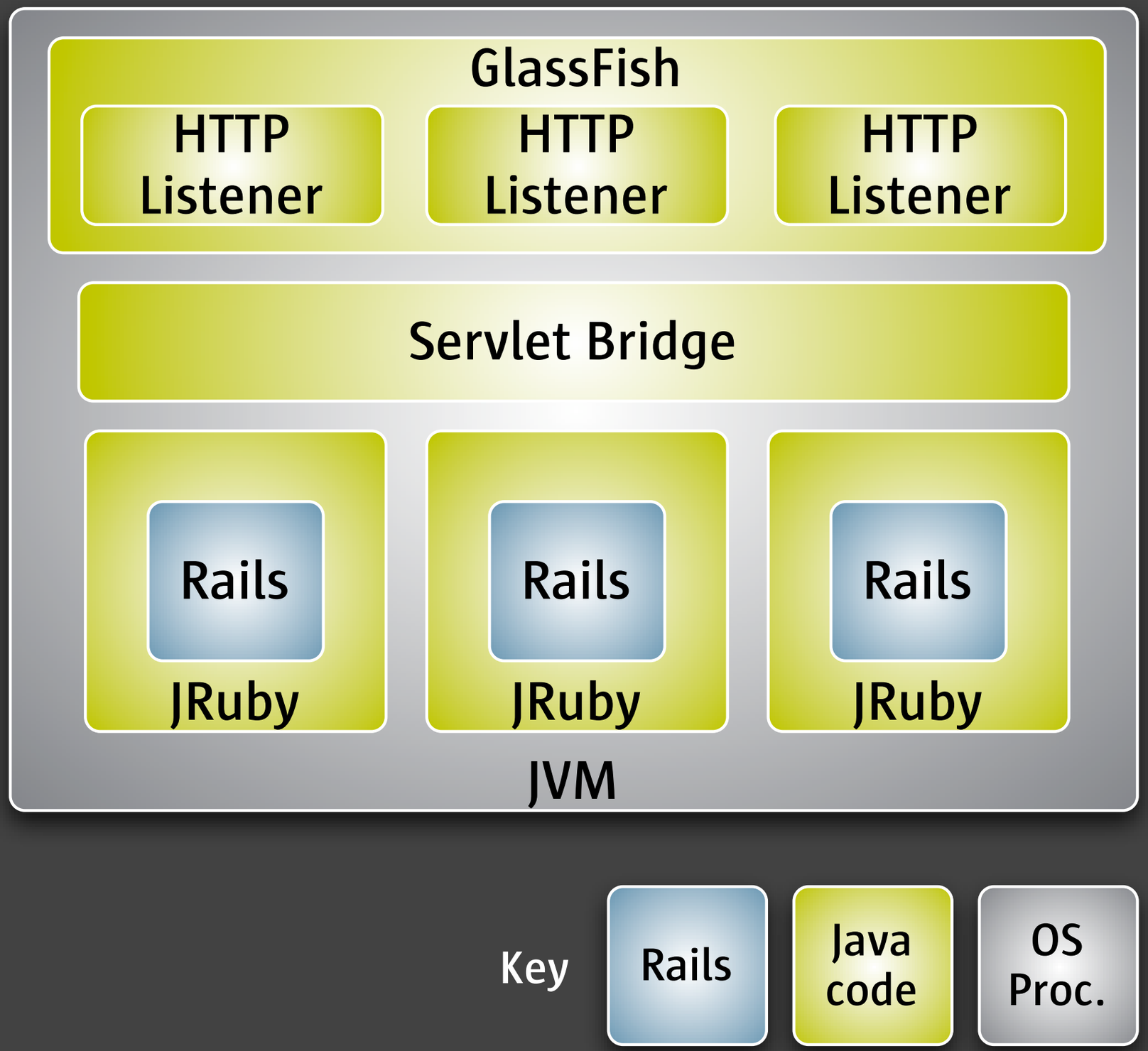
- Config file is written in Ruby
- Easily write your own custom conditions in Ruby

```
Processes:  68 total, 3 running, 1 stuck, 64 sleeping... 309 threads
Load Avg:  0.71,  0.51,  0.43    CPU usage: 22.79% user, 13.02% sys, 64.19% idle
SharedLibs: num =    8, resident =   80M code,  384K data, 5156K linkedit.
MemRegions: num = 13190, resident =  707M +   27M private,  400M shared.
PhysMem:  589M wired, 1205M active,   89M inactive, 1883M used, 2197M free.
VM: 11G + 373M   188123(0) pageins, 0(0) pageouts

  PID COMMAND      %CPU   TIME   #TH #PRTS #MREGS RPRVT  RSHRD  RSIZE  VSIZE
  650 top          6.3%  0:01.71   1    18     29 1084K   188K  1676K    18M
  571 ruby         0.2%  0:04.09   2    17    163   32M   188K    34M    51M
  572 ruby         0.2%  0:04.09   2    17    163   32M   188K    34M    51M
  573 ruby         0.2%  0:04.09   2    17    163   32M   188K    34M    51M
  574 ruby         0.2%  0:04.09   2    17    163   32M   188K    34M    51M
  575 ruby         0.2%  0:04.09   2    17    163   32M   188K    34M    51M
  576 ruby         0.2%  0:04.09   2    17    163   32M   188K    34M    51M
  577 ruby         0.2%  0:04.09   2    17    163   32M   188K    34M    51M
  578 ruby         0.2%  0:04.09   2    17    163   32M   188K    34M    51M
  579 ruby         0.2%  0:04.09   2    17    163   32M   188K    34M    51M
  580 ruby         0.2%  0:04.09   2    17    163   32M   188K    34M    51M
  581 ruby         0.2%  0:04.09   2    17    163   32M   188K    34M    51M
  582 ruby         0.2%  0:04.09   2    17    163   32M   188K    34M    51M
  583 ruby         0.2%  0:04.09   2    17    163   32M   188K    34M    51M
  584 ruby         0.2%  0:04.09   2    17    163   32M   188K    34M    51M
  585 ruby         0.2%  0:04.09   2    17    163   32M   188K    34M    51M
  586 ruby         0.2%  0:04.09   2    17    163   32M   188K    34M    51M
  587 ruby         0.2%  0:04.09   2    17    163   32M   188K    34M    51M
  588 ruby         0.2%  0:04.09   2    17    163   32M   188K    34M    51M
  589 ruby         0.2%  0:04.09   2    17    163   32M   188K    34M    51M
  580 ruby         0.2%  0:04.09   2    17    163   32M   188K    34M    51M
  591 ruby         0.2%  0:04.09   2    17    163   32M   188K    34M    51M
```


Webserver/Load Balancer
Mongrel
Rails
Ruby
Mongrel
Rails
Ruby
Mongrel
Rails
Ruby


Key
Rails
Ruby code
OS Proc.


GlassFish
HTTP Listener
HTTP Listener
HTTP Listener
Servlet Bridge
Rails
JRuby
Rails
JRuby
Rails
JRuby
JVM
Key
Rails
Java code
OS Proc.

existing applications

```
$ jruby -S gem install activerecord-jdbcmysql-adapter
Successfully installed activerecord-jdbc-adapter-0.8
Successfully installed jdbc-mysql-5.0.4
Successfully installed activerecord-jdbcmysql-adapter-0.8
3 gems installed
Installing ri documentation for activerecord-jdbc-adapter-0.8...
Installing ri documentation for jdbc-mysql-5.0.4...
Installing ri documentation for activerecord-jdbcmysql-adapter-0.8...
Installing RDoc documentation for activerecord-jdbc-adapter-0.8...
Installing RDoc documentation for jdbc-mysql-5.0.4...
Installing RDoc documentation for activerecord-jdbcmysql-adapter-0.8...
```

```
# config/database.yml
<% jdbc = defined?(JRUBY_VERSION) ? 'jdbc' : '' %>
development:
  adapter: <%= jdbc %>mysql
  encoding: utf8
  database: testapp_development
  username: root
  password:
  socket: /tmp/mysql.sock

# same for test/production...
```

Gems and libraries with native code

http://flickr.com/photos/thehappyrobot/1413968365/

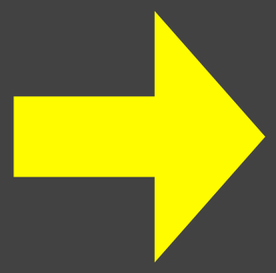
replacements/ equivalents

Mongrel, Hpricot OK

RMagick,
ImageScience

ImageVoodoo

OpenSSL



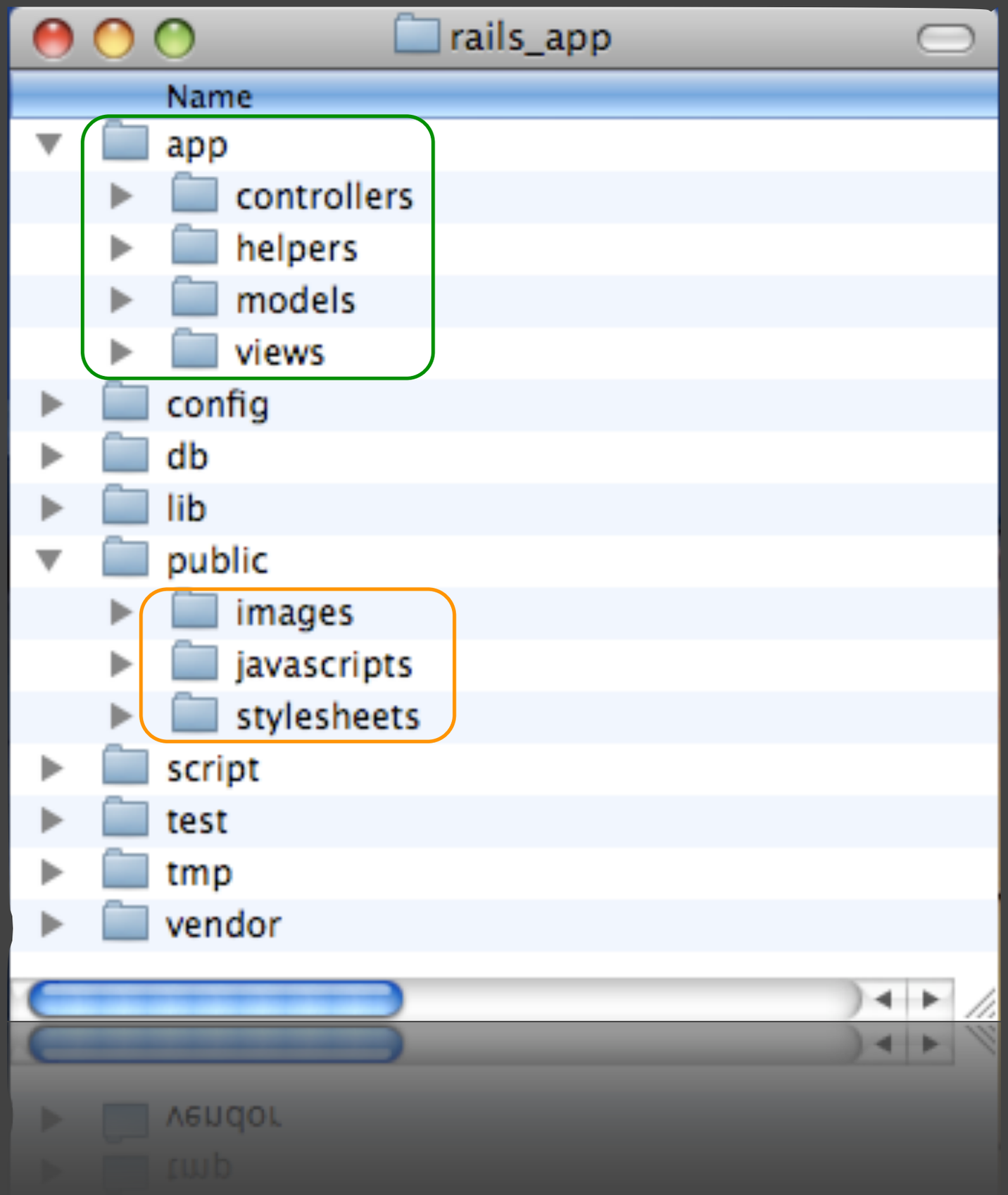
JRuby-
OpenSSL
gem

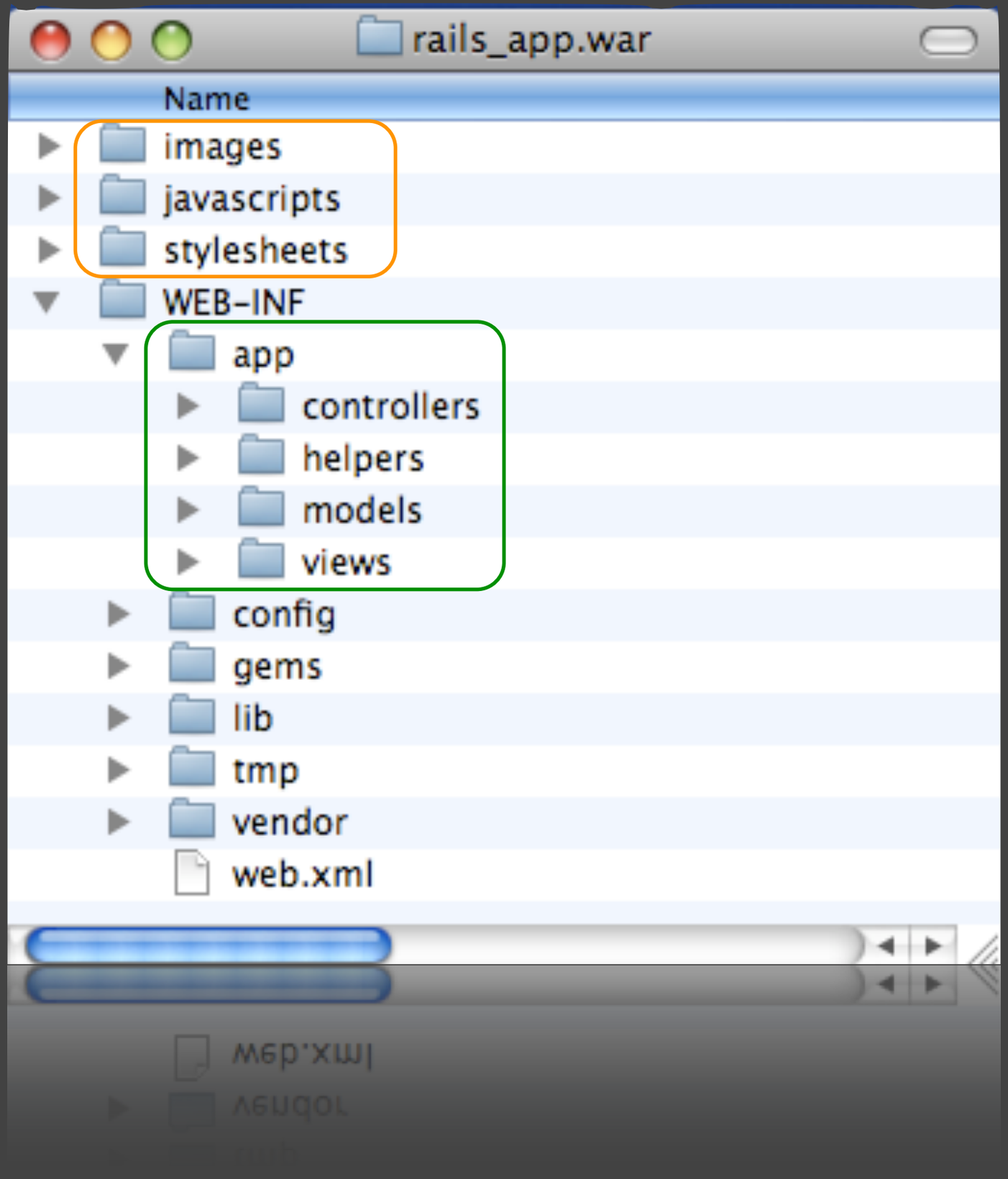
Ruby/LDAP

JRuby/LDap

```
$ jruby -S gem install mongrel
Successfully installed mongrel-1.1.3-java
1 gem installed
Installing ri documentation for mongrel-1.1.3-java...
Installing RDoc documentation for mongrel-1.1.3-java...
$ jruby script/server
=> Booting Mongrel (use 'script/server webrick' to force WEBrick)
=> Rails application starting on http://0.0.0.0:3000
=> Call with -d to detach
=> Ctrl-C to shutdown server
** Starting Mongrel listening at 0.0.0.0:3000
** Starting Rails with development environment...
** Rails loaded.
** Loading any Rails specific GemPlugins
** Signals ready.  TERM => stop.  USR2 => restart.  INT => stop (no restart).
** Rails signals registered.  HUP => reload (without restart).  It might not work
well.
** Mongrel 1.1.3 available at 0.0.0.0:3000
** Use CTRL-C to stop.
```

packaging


rails_app
Name
app
controllers
helpers
models
views
config
db
lib
public
images
javascripts
stylesheets
script
test
tmp
vendor


rails_app.war
Name
images
javascripts
stylesheets
WEB-INF
app
controllers
helpers
models
views
config
gems
lib
tmp
vendor
web.xml

warbler

```
jruby -S gem install warbler
jruby -S warble config
```

```
# Warbler web application assembly configuration file
Warbler::Config.new do |config|
  # ...

  # Gems to be packaged in the webapp.  ...
  config.gems += ["activerecord-jdbcmysql-adapter",
    "jruby-openssl"]
  config.gems["rails"] = "2.0.2"

  # ...
end
```

warbler demo

configuration

logging

```
if defined?(JRUBY_VERSION) && defined?($servlet_context)
  # Logger expects an object that responds to #write and #close
  device = Object.new
  def device.write(message)
    $servlet_context.log(message)
  end
  def device.close; end

  # Make these accessible to wire in the log device
  class << RAILS_DEFAULT_LOGGER
    public :instance_variable_get, :instance_variable_set
  end

  old_device = RAILS_DEFAULT_LOGGER.instance_variable_get "@log"
  old_device.close rescue nil
  RAILS_DEFAULT_LOGGER.instance_variable_set "@log", device
end
```

sessions

```
config.action_controller.session_store = :java_servlet_store

# ...

class CGI::Session::JavaServletStore
  def initialize(session, options) end
  def restore; end
  def update; end
  def close; end
end
```

connection pools

```
# config/initializers/close_connections.rb
if defined?($servlet_context)
  require 'action_controller/dispatcher'
  ActionController::Dispatcher.after_dispatch do
    ActiveRecord::Base.clear_active_connections!
  end
end
```

view caching

```
config.action_view.cache_template_loading = true
```

(default in Rails >= 2.0.2)

asset timestamps

```
ENV['RAILS_ASSET_ID'] = "r#{source_revision}"
```

full-page cache

```
config.action_controller.page_cache_directory =
"/where/is/my/railsapp/war/deployed"
```

Rails.public_path coming in 2.1

```ruby
if defined?(JRUBY_VERSION) && defined?($servlet_context)
  # Logger expects an object that responds to #write and #close
  device = Object.new
  def device.write(message)
    $servlet_context.log(message)
  end
  def device.close; end

  # Make these accessible to write in the logs below
  class << RAILS_DEFAULT_LOGGER
    public :instance_variable_get, :instance_variable_set
  end
  old_device = RAILS_DEFAULT_LOGGER.instance_variable_get "@log"
  old_device.close rescue nil
  RAILS_DEFAULT_LOGGER.instance_variable_set "@log", device
end
```

```ruby
config.action_controller.session_store = :java_servlet_store
# ... # config/initializers/close_connections.rb
if defined?($servlet_context)
  require 'action_controller/dispatcher'
  config.action_controller.cache_template_loading = true
  ENV['RAILS_ASSET_ID'] = "#{source_revision}"
  RAILS_DEFAULT_LOGGER = "/where/els/my/railsapp/war/deployed"
  ActiveRecord::Base.clear_active_connections!
end
```

FAIL

jruby-rack

available now

released earlier this month

bundled with warbler

servlet adapter

eliminates previous configuration

http://rack.rubyforge.org/

```
proc do |env|
  [200, {"Content-Type" => "text/html"}, "Hello World!"]
end
```

“middleware”

```ruby
module Rack
  class ShowStatus
    def initialize(app)
      @app = app
      @template = ERB.new(TEMPLATE)
    end

    def call(env)
      status, headers, body = @app.call(env)
      if status.to_i >= 400
        [status,
          headers.merge("Content-Type" => "text/html"),
          [@template.result(binding)]]
      else
        [status, headers, body]
      end
    end
  end
end
```

rack demo

tuning

http://flickr.com/photos/86974734@N00/2206011624/

pool size

```
Warbler::Config.new do |config|
  # ...
  # Control the pool of Rails runtimes
  config.webxml.jruby.min.runtimes = 2
  config.webxml.jruby.max.runtimes = 4
end
```

~~performance~~

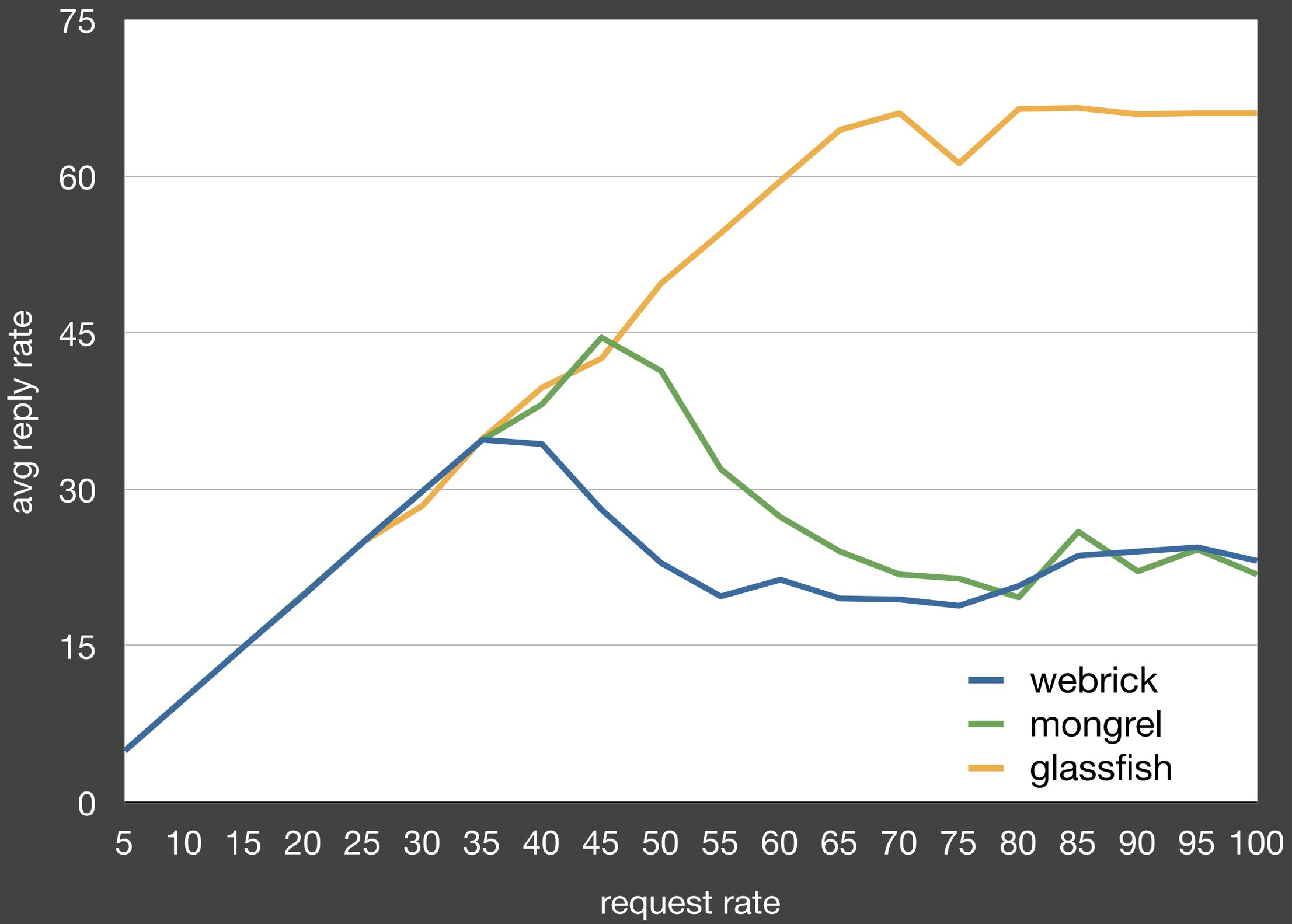
misattribution

lies, damned lies

YMMV

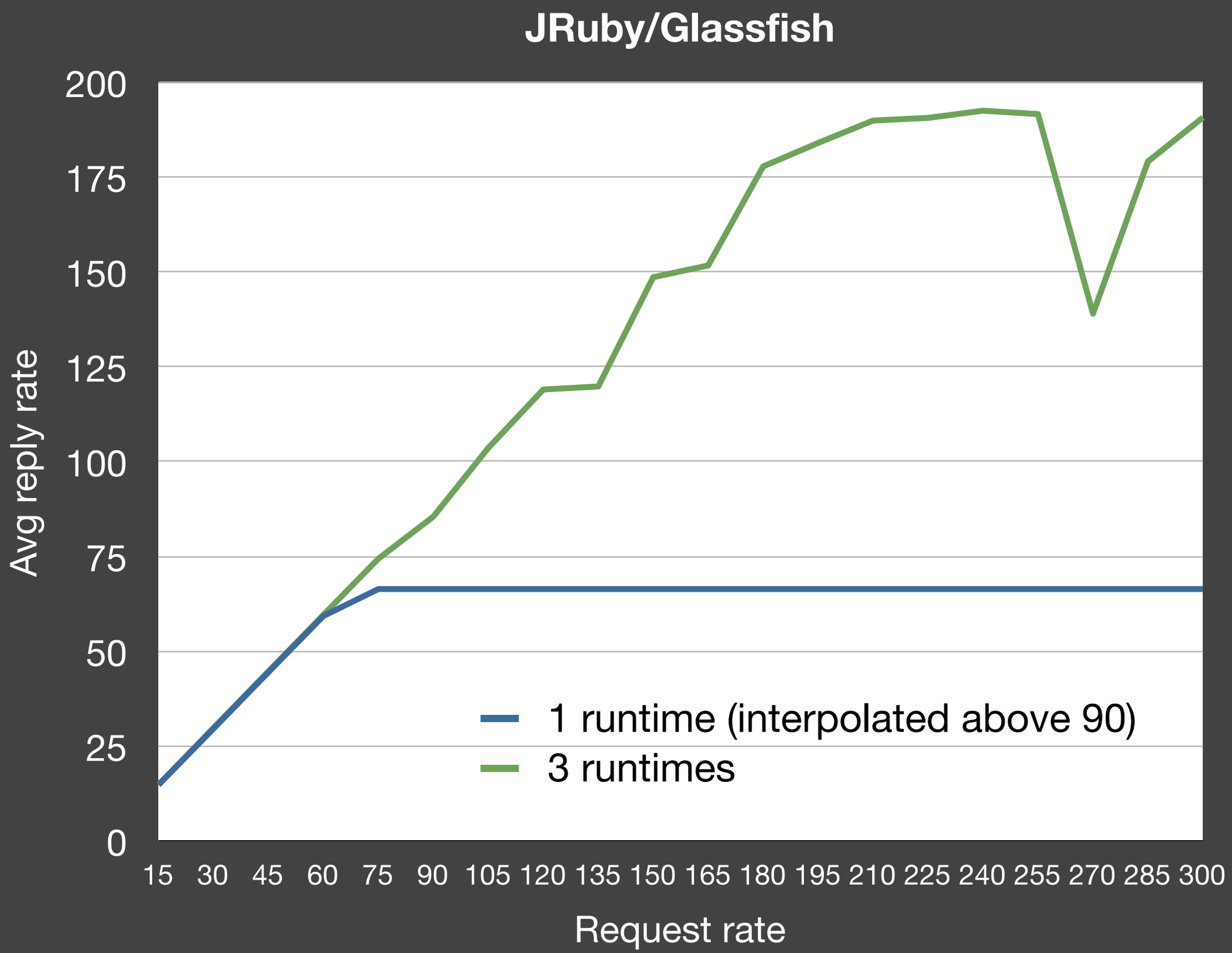
measure your own @#$!& app

Request Scaling (1 process or runtime)



```
Warbler::Config.new do |config|
  # ...
  # Control the pool of Rails runtimes
  config.webxml.jruby.min.runtimes = 1
  config.webxml.jruby.max.runtimes = 1
end
```

```
Warbler::Config.new do |config|
  # ...
  # Control the pool of Rails runtimes
  config.webxml.jruby.min.runtimes = 3
  config.webxml.jruby.max.runtimes = 3
end
```


JRuby/Glassfish
Avg reply rate
200
175
150
125
100
75
50
25
0
15 30 45 60 75 90 105 120 135 150 165 180 195 210 225 240 255 270 285 300
Request rate
1 runtime (interpolated above 90)
3 runtimes

the future is now

rails concurrency

Each Rails request must execute in isolation.

“Rails is not thread-safe? WTF?”

“Servlets have been multi-threaded since, like,1997!”

concurrency
context

Writing thread-safe code is hard

\+ Ruby's userland threads don't yield much benefit

Not worth the effort

but what if it was?

Rails to become thread-safe

(Google SoC 2008 project)

connection pooling for ActiveRecord

(yours truly)

Rails on a Rack?

Ezra’s work: http://github.com/ezmobius/rails/

rails
rack
native threading
connection pooling
single-file deployment
+ industrial-strength vm

“she’s a beaut.”

hassle-free

scalable

(enterprisey?)

awesome!

(stay tuned!)