Comparing the Developer Experience of Java™ EE 5.0, Ruby on Rails, and Grails

Tom Daly, Senior Performance Engineer
Damien Cooke, ISV Engineering

Sun Microsystems, Inc.
http://www.sun.com

TS-9535

Goal: Compare 3 Increasingly Popular Development Frameworks

Introduce developers to each framework and highlight some of the strengths and weakness of each.

Help developers decide where to invest their time/expertise.

Agenda

Background

Why Ruby on Rails, Grails, and Java EE 5.0

Our Application “RegPerf”

Performance

Observations (What’s Cool, What’s Not)

Resources

Ruby Background

What it is, where did it come from

- Ruby history
- jRuby
- Tools
- How does Ruby on Rails emerge from Ruby?
- What is Sun doing with Ruby/jRuby?
- Popularity (adoption)

Ruby on Rails Application Structure

What makes Rails tick

- Controllers
- Models
- Views
- Active Record
- Webbrick

Rails Application Structure

Generate the subscription controller

```
class SubscriptionsController < ApplicationController

  scaffold :subscription
end
```

- #script/generate controller subscriptions
- Add the line `scaffold :subscription`

Rails Application Structure

Generate the model

```
class Subscription < ActiveRecord::Base

end
```

- #script/generate model subscription
- The rest is automatically generated from the database table by the plural of the model name

JavaOne

Grails Background

What is Grails, history

- Grails is based on Groovy
  - Groovy is a JavaScript™ programming language
  - Integrates with Java technology
  - Java Specification Request (JSR) 241, JSR 223
- Ruby on Rails inspired Groovy on Rails
  - But had to change the name, hence, Grails
- Grails integrates with Java Platform, Enterprise Edition (Java EE) 5.0 and application servers, "**grails war**" produces a Java EE 5.0 deployable Grails application

Grails Application Structure

Domain objects (c.f. Java Persistence API [JPA] Entity)

Subscriber.groovey

```
class Subscriber {
    String firstName;
    String lastName;
    String address1;
    ..
}
```

- Grails **generate-all** generates the controller and the views

Grails Application Structure

Controller

```
class SubscriberController {
    def index = { redirect(action:list,params:params) }
    def allowedMethods = [delete:'POST',
                          save:'POST',
                          update:'POST']
    def list = {
        if(!params.max)params.max = 10
        [ subscriberList: Subscriber.list(
            params ) ]
    }

    def show = {
        [ subscriber : Subscriber.get( params.id ) ]
    }
    ....
```

Grails Application Structure

Views, e.g., the table rendering snippet from view.gsp

```
<tbody>
  <g:each in="${subscriberList}">
     <tr>
        <td>${it.id?.encodeAsHTML()}</td>
        <td>${it.address?.encodeAsHTML()}</td>
          ...


        <td class="actionButtons">
           <span class="actionButton"><g:link
action="show" id="${it.id}">Show</g:link></span>

   </g:each>
</tbody>
```

Java EE 5.0

Are you kidding? This is JavaOne[SM]!

Agenda

Background

Why Ruby on Rails, Grails, and Java EE 5.0

Our Application “RegPerf”

Performance

Observations (What’s Cool, What’s Not)

Resources

Why Ruby on Rails, Grails, and Java EE 5.0

How did we get here?

- Why do these “new” frameworks exist?
- Are they growing in popularity and why?
  - Tools support happening
  - Books, conferences, and more
- Learning curve and design point for Java 2 Platform, Enterprise Edition (J2EE™ platform) 1.4
- “Innovation happens elsewhere” (Bill Joy)
- Simplicity for RAD web applications

Why Ruby on Rails, Grails, and Java EE 5.0

But the biggest drivers are probably:

- Simplicity
  - Very easy to develop web applications
- Speed
  - It only takes a very small time to create useful and powerful web applications

DEMO

Just How Fast Can You Build a Simple Web Application With Ruby on Rails?

Agenda

Background

Why Ruby on Rails, Grails, and Java EE 5.0

Our Application "RegPerf"

Performance

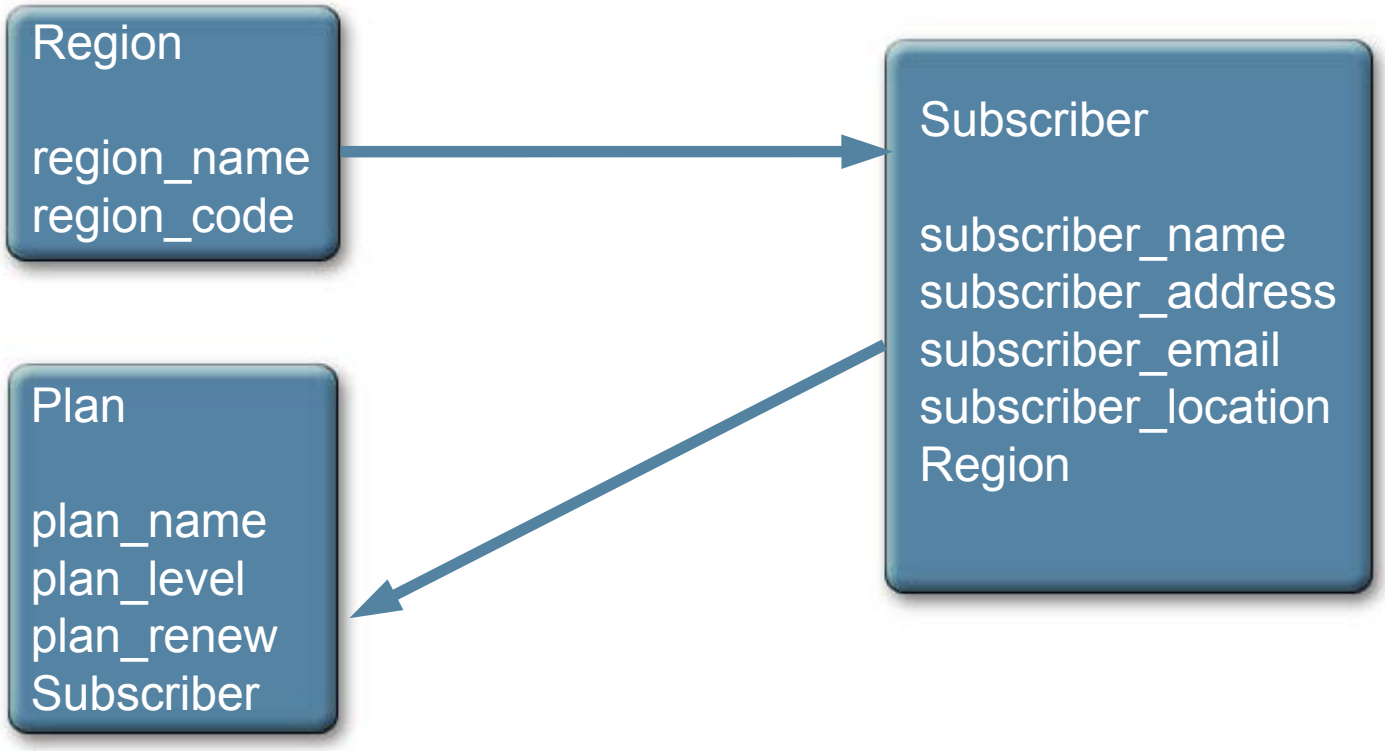
Observations (What's Cool, What's Not)

Resources

Our Demo Application RegPerf

Software registration performance demo application

- Models a software support registration system



Our Demo Application RegPerf (Cont.)

Why is this application relevant?

- You are using elements of this application in your business
  - Create, Update, Delete data via web application
  - e.g., Payroll, HR, and Help desk
  - …or even user registration systems
- Because you can use this for performance testing in your organization

Building the Application in RoR

Using NetBeans™ IDE 6.0, RoR modules, and JRuby

- Build database tables
- Edit config for Java DataBase connectivity (JDBC™) and database
- Generate controllers
- Generate models
- Run application

Building the Application Grails

Using vi + ksh

- Build the application
  - **Grails create-application** RegPerf
- Connect application to database
  - Edit RegPerf/grails-app/conf/*DataSource.groovy
    - Copy the JDBC software driver to RegPerf/grails-app/lib

```
class DevelopmentDataSource {
   boolean pooling = true
   String dbCreate = "create-drop"
   String url = "jdbc:postgresql://pdb:5432/gregperfdb"
   String driverClassName = "org.postgresql.Driver"
   String username = "tdaly"
   String password = "tdaly"
```

Building the Application Grails

Using vi + ksh

- Define the domain objects (Groovy objects)
  - Edit RegPerf/grails-app/domain/Subscriber.groovy, Plan.groovy and Location.groovy
- Generate controllers,views, and rest of application
  - **Grails generate-all Subscriber**

Building the Application in Java EE 5.0

Using NetBeans IDE

- Build the entities
- Create the persistance.xml
- Generate JavaServer™ Faces CRUD application
- Or generate entities from DB schema

- Application building can start with Java class files or existing database schema or object model

DEMO

Building the RegPerf Application With Grails and Ruby on Rails and Java EE 5.0

Agenda

Background
Why Ruby on Rails, Grails, and Java EE 5.0
Our Application “RegPerf”
Performance
Observations (What’s Cool, What’s Not)
Resources

Performance Experiments

Across the three frameworks

- Comparing performance of Ruby and JRuby
- Implications on the database
- Webbrick or Apache or Project GlassFish™/ Java EE 5.0?

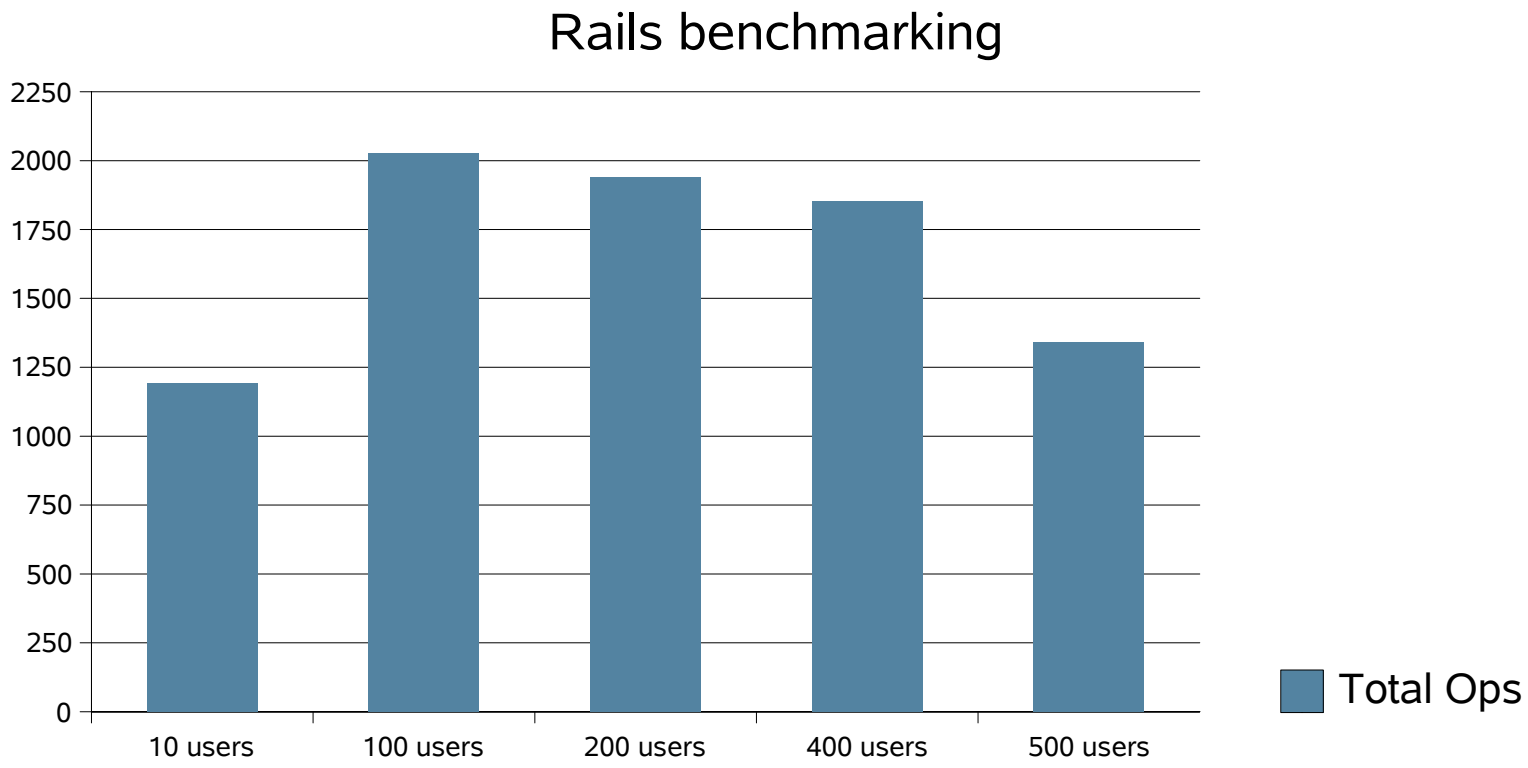
Performance

Performance testing

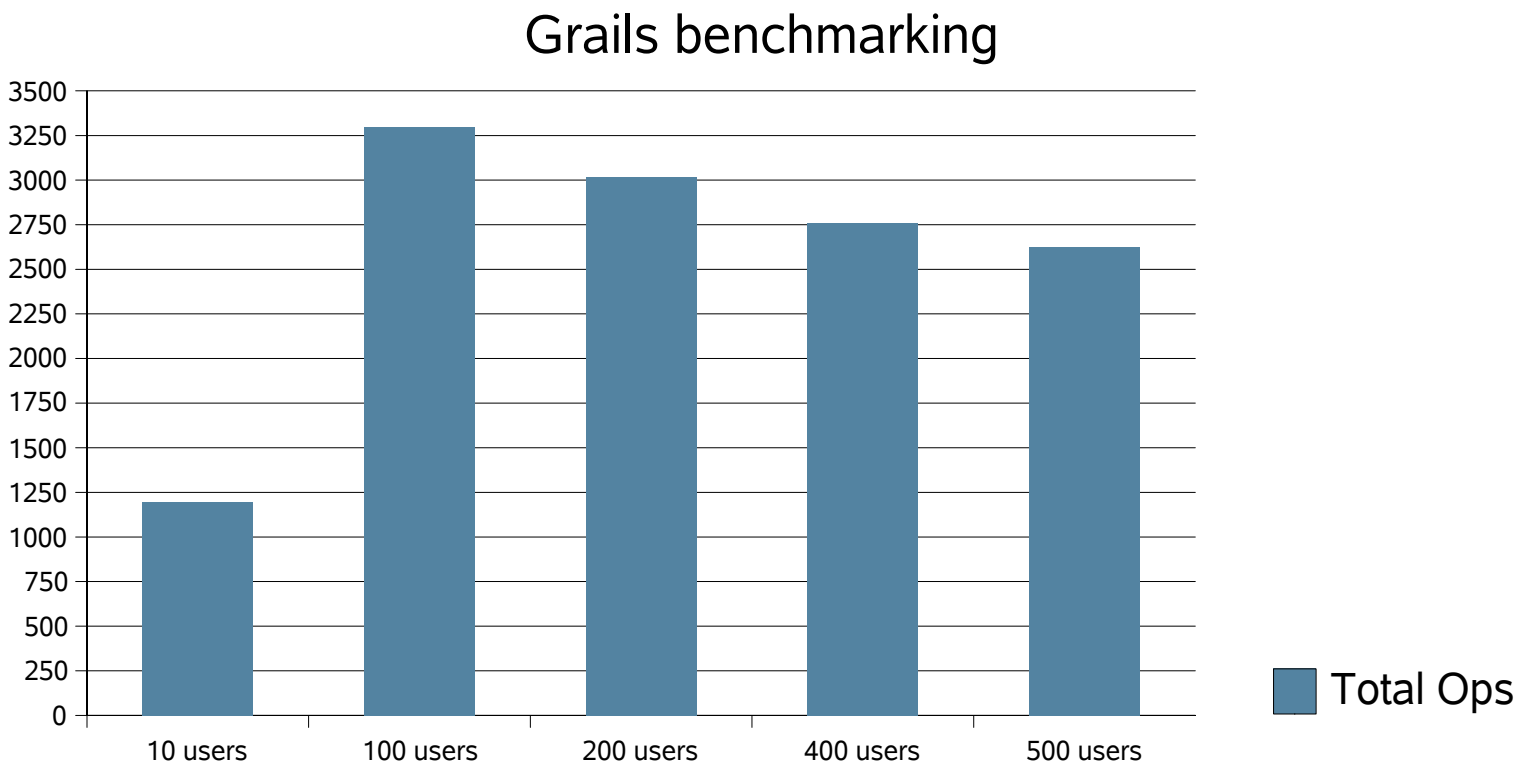
- Using FABAN (http://faban.sunsource.net)

- Generate user population performing CRUD operations on each framework/environment and compare throughput response time and CPU util

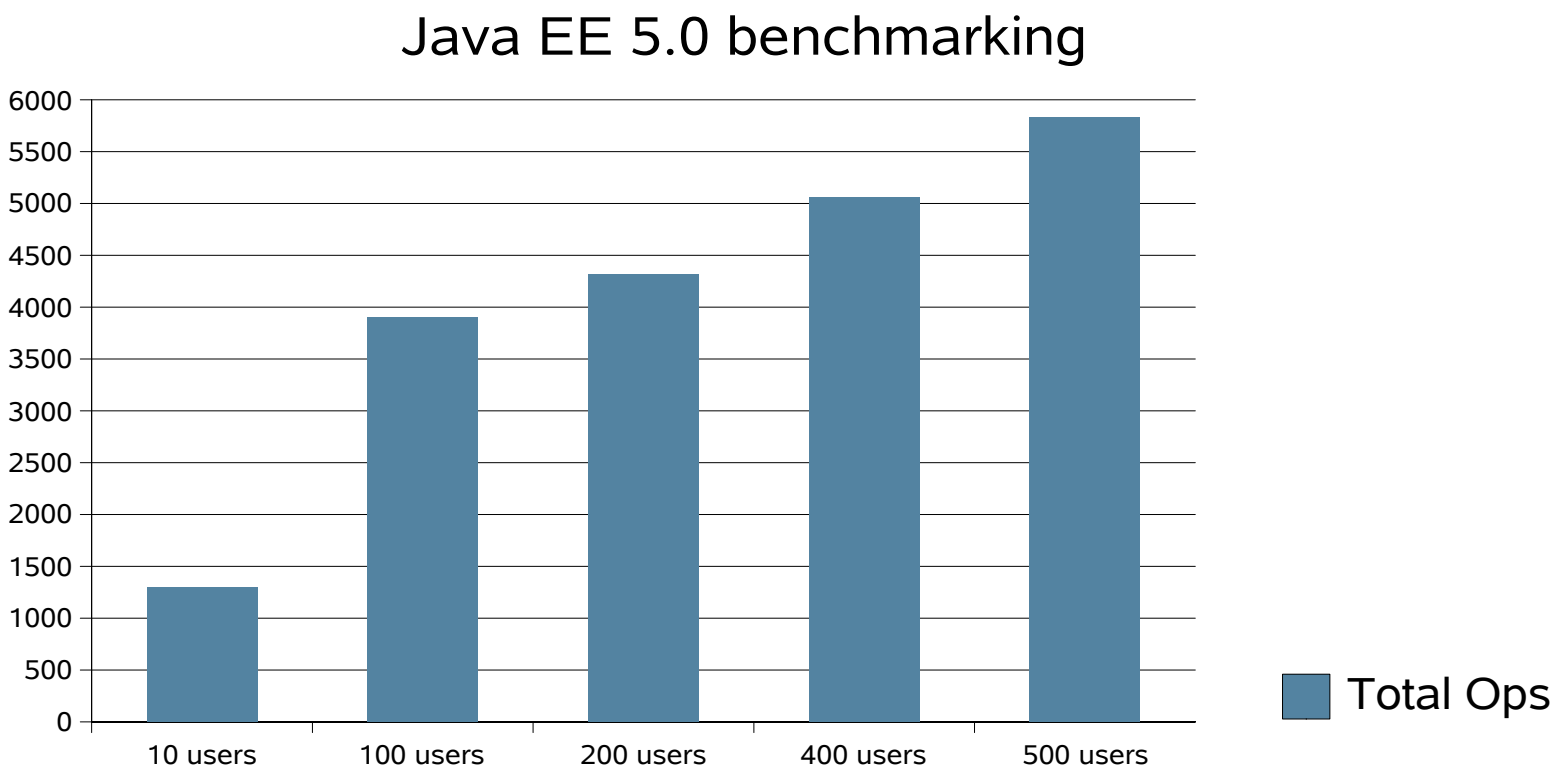
RegPerf Testing Results: Rails



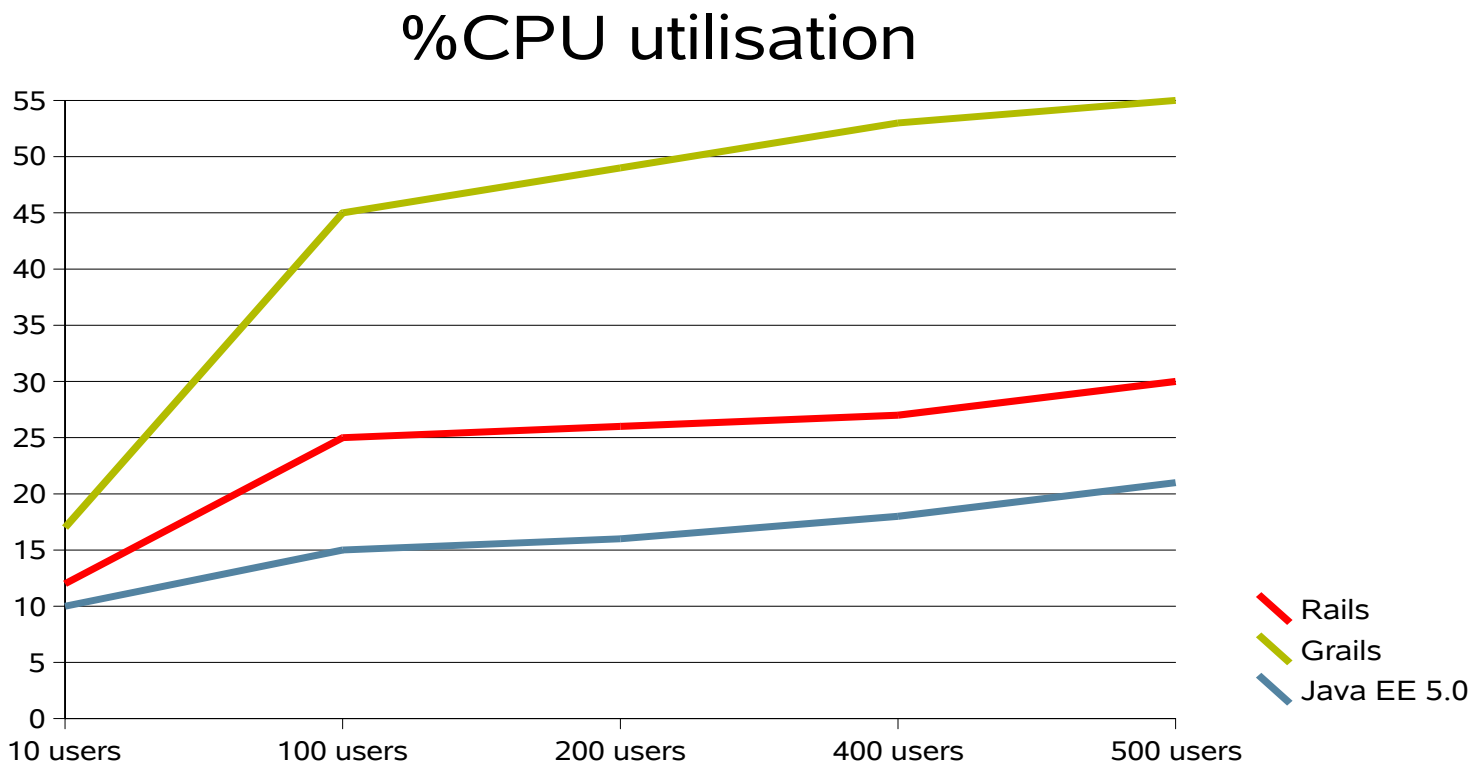
RegPerf Testing Results: Grails



RegPerf Testing Results: Java EE 5.0



RegPerf Testing Results



Agenda

Background
Why Ruby on Rails, Grails, and Java EE 5.0
Our Application “RegPerf”
Performance
Observations (What’s Cool, What’s Not)
Resources

From Rails on Jruby to Rails on Ruby

- Native Ruby outperforms Jruby at this stage
- NetBeans IDE could be the best-of-breed dev platform for Rails apps
- So I wrote in NetBeans IDE/JRuby and deployed on native Ruby
- Changes needed to make the NetBeans IDE project run on native Ruby
  - config/database.yml
  - Install ruby postgres-pr bridge
  - config/boot.rb (Ruby version)

Ruby on Rails: Cool Stuff

- How long did it take to actually get useful application running?
- What about making the application robust?
- Tools

Configuring Rails on Solaris™ Operating System (Solaris™ OS) 10

- The gem script may need a change, else you may get the following error:
- /usr/bin/env no such file?

- #!/usr/bin/env ruby
- #!/usr/local/bin/ruby
  (This could be an installation anomaly but it happened on both of my machines)

Ruby on Rails: Not So Cool Stuff

- Don’t touch the database once app is created!

Grails: Cool Stuff!

- Incredibly short learning curve
  - Having a useful template (scaffold) is very powerful
  - Productive even without knowing Groovy
  - Following simple examples is surprisingly productive
  - Easy taglib integration
  - Grails OR Mapping **GORM** is easy to understand for Java EE software programmers

Grails: Cool Stuff! (Cont.)

- Integration with Java EE 5.0
  - Grails .war deploys to Project GlassFish, Java EE platform servers
  - Grails can use EJB™ 3.0 beans for persistence today
    - But it deploys Hibernate + everything to the Java EE platform server
    - Note may require **-XX:MaxPermSize=128m** or more
  - Will be able to use JPA for persistence in Grails 1.0!

Grails: Not So Cool Stuff

- Technologies from everywhere
  - Hibernate, GORM, GS , Spring, Groovy...
    - To do customization, eventually have to know some of these
- Changes to the domain classes problematic
  - Not clear how to regenerate if you have made changes to the controller/views
  - Some limitations on updating DB schema
- Documentation is still “light”
  - Web site codehaus.org is very good
  - Transaction support not well documented
- Little tools support (yet)

DEMO

Deploying a Grails Application to Project GlassFish, Java EE 5.0, and Postgres DB

Java EE 5.0: Cool Stuff

- Familiarity
  - Years of experience from EJB beans, J2EE technology, JavaServer™ Pages (JSP™) technology
  - Taught in Universities world wide
- Java EE 5.0 is far simpler than J2EE platform 1.4
  - Except perhaps for the richness of JavaServer Faces technology
- Ability to generate JavaServer Faces CRUD application from entities using NetBeans IDE

Java EE 5.0: Cool Stuff (Cont.)

- Scalability/performance
  - Well understood for Java EE, caching understood
  - SPECjAppServer 2004 Benchmark
- Strong transaction support
- Great tools for Java EE 5.0
- Standards

Java EE 5.0: Not So Cool

- A simple CRUD application with JSTL/JSP technology is much more work
- Do have to do more learning before you start

Conclusions/Observations

RoR/Grails Frameworks great for CRUD/RAD, but there are still some questions:

- Scalability
  - Large datasets
  - Large user populations
  - Complex applications
  - Data concurrency considerations
- Maintainability
  - Large code bases
  - Customizations

Conclusions/Observations

RoR/Grails Frameworks great for CRUD/RAD, but there are still some questions:

- Standards
  - What standards will protect your investment
  - Portability
- Support
- Adoption

Summary

- We found that for many web applications RoR and Grails are simple yet powerful tools and are likely to continue to grow in popularity
- Java EE 5.0 will also continue to grow due to:
  - Performance
  - Industry backing
  - Scalability
  - Standards, portability…
- So choose the tool according to the application

Agenda

Background
Why Ruby on Rails, Grails, and Java EE 5.0
Our application “RegPerf”
Performance
Observations (What’s Cool, What’s Not)
Resources

Next Steps (What We Would Like to See)

- Continuing the development of RegPerf
  - Extend benchmark to include transaction support using each framework's default transaction mechanism
  - Study the effects of data contention
- Scale up
  - Test best possible configurations
  - Develop some best practices
- Community
  - Figure out how to enable others to contribute

Resources

- blogs.sun.com/damien/rails
- blogs.sun.com/tomdaly/grails
- blogs.sun.com/damien/regperf
- blogs.sun.com/tomdaly/regperf
- rubyonrails.org
- codehaus.org
- faban.sunsource.net/

Q&A

Comparing the Developer Experience of Java™ EE 5.0, Ruby on Rails, and Grails

Tom Daly, Senior Performance Engineer
Damien Cooke, ISV Engineering

Sun Microsystems, Inc.
http://www.sun.com

TS-9535