



Scripting Na JVM

Maior Produtividade Para a Plataforma Java?

Miguel Duarte
malduarte@gmail.com
2008-05-29



Vantagens do Java

- Está por (quase) todo o lado
- Bom desempenho
- Excelentes ferramentas de desenvolvimento (comerciais ou não)
- Inúmeras bibliotecas disponíveis (comerciais ou não)
- Mercado amplo
- Boa *pool* de recursos disponíveis

Fazer uma breve abordagem às vantagens do java



Linguagem Vs Plataforma

Java Language	Java Language									
Tools & Tool APIs	java	javac	javadoc	apt	jar	javap	JPDA	jconsole		
	Security	Int'l	RMI	IDL	Deploy	Monitoring	Troubleshoot	Scripting	JVM TI	
Deployment Technologies	Deployment			Java Web Start			Java Plug-in			
User Interface Toolkits	AWT			Swing			Java 2D			
	Accessibility	Drag n Drop		Input Methods		Image I/O	Print Service	Sound		
Integration Libraries	IDL	JDBC™	JNDI™		RMI	RMI-IIOP		Scripting		
Other Base Libraries	Beans	Intl Support		I/O	JMX	JNI		Math		
	Networking	Override Mechanism		Security	Serialization	Extension Mechanism		XML JAXP		
lang and util Base Libraries	lang and util	Collections	Concurrency Utilities		JAR		Logging	Management		
	Preferences API	Ref Objects	Reflection		Regular Expressions		Versioning	Zip	Instrument	
Java Virtual Machine	Java Hotspot™ Client VM					Java Hotspot™ Server VM				
Platforms	Solaris™			Linux		Windows		Other		

Java SE API

Java é muito mais que a linguagem. Deve ser entendido como uma plataforma em que a linguagem Java apenas é uma das formas a que podemos aceder a todos os outros componentes da plataforma (bibliotecas, a VM, etc)



Sinais de alarme

- A linguagem estagnou!
- Frameworks de desenvolvimento com curvas de aprendizagem longas
- As inovações mais interessantes aparecem fora do mundo Java!
- Tempo de desenvolvimento “longo” – Arquitecturas complexas

Apesar das vantagens da Plataforma Java, não é a linguagem de escolha para o desenvolvimento na web em pequena média escala



Os Astronautas da Arquitectura

(...)When you go too far up, abstraction-wise, you run out of oxygen. Sometimes **smart thinkers** just don't know when to stop, and they create these **absurd, all-encompassing, high-level pictures of the universe** that are all good and fine, but don't actually mean anything at all. (...)

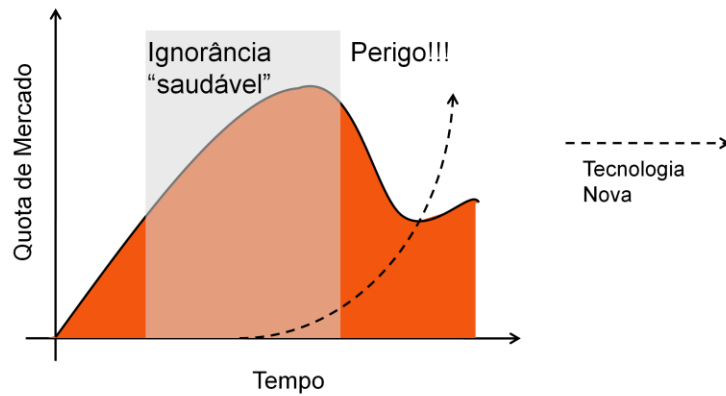
Joel Spolsky 2004-04-21 – Joel On Software

<http://www.joelonsoftware.com/articles/fog0000000018.html>

Ideia chave: A abstracção não é um fim em si. A abstracção deve contribuir para reduzir o fosso semântico do desenvolvimento, não o contrário



A ingorância



Adaptado de "Beyond Java" Bruce A. Tate, O'Reilly - 2005

Ideia chave: não podemos dominar todas as tecnologias que surgem, mas convém estarmos atentos aos sinais de mudança



Características do “scripting”

- Reduz o fosso semântico
- Estruturas de dados frequentes embutidas na sintaxe (listas, arrays, maps)
- Expressões Regulares
- Closures
- Feedback imediato (interpretadas ou não)
- Iteração simplificada
- “Loosely typed” (não necessariamente)

Falar da ideia de fosso semântico : “Distância” entre o problema que estamos a resolver e a implementação da solução.



Caso de Estudo – Projecto Wide Finder

<http://www.tbray.org/ongoing/When/200x/2007/09/20/Wide-Finder>

- Qual a forma mais rápida de analisar logs
- Ficheiro de 927 Mb, 4.625.236 linhas

fatwire-202-11.uniserve.ca - - [26/Feb/2007:17:53:20 -0800]

"GET /ongoing/When/200x/2007/02/26/Netbeans-6 HTTP/1.1" 200 7251

"http://www.tbray.org/ongoing/" "Mozilla/5.0 (Macintosh; U; PPC Mac OS X Mach-O; en; rv:1.8.1.2pre) Gecko/20070223Camino/1.1b"

- Os 10 urls mais visitados

40173	2006/03/13/Rock-n-Roll-Animal
7570	2006/03/30/Teacup
7407	2006/03/17/Church
6635	2007/02/25/High-End-Compact-Cameras
4737	2007/02/26/Netbeans-6
4676	2007/02/24/OpenID
2398	2007/02/22/Economist
2305	2003/04/26/UTF
2299	2007/02/21/Comments
2171	2007/02/27/Udell-Vinowski

Introduzir o WideFinder como um desafio simpático para avaliar as funcionalidades de uma linguagem, desempenho e benchmarking e quebrar alguns “mitos”

- Problema : Obter o top 10 dos urls mais visitados a partir do log de um webserver
- Realçar que o objectivo original do WideFinder era o de estudar qual a linguagem mais adequada para tirar partido de sistemas multi-core. Neste caso apenas foi aproveitado para ilustrar as vantagens do scripting

Analisar a linha exemplo de log



Implementação Java

```
1 import java.io.*;
2 import java.util.*;
3 import java.util.Map.Entry;
4 import java.util.regex.*;
5
6 public class WfReader {
7     public static void main(String[] args) throws IOException {
8         BufferedReader br = new BufferedReader(new FileReader(args[0]));
9         Map<String, Integer> urls = new HashMap<String, Integer>();
10        String linha;
11        Pattern p = Pattern.compile("GET /ongoing/When/\\d(3)x/\\d(4)/\\d\\d/\\d\\d/[^\"]+ ");
12        Matcher m = p.matcher("");
13
14        while((linha = br.readLine()) != null) {
15            m.reset(linha);
16            if(m.find()) {
17                String url = m.group(1);
18                Integer n = urls.get(url);
19                urls.put(url, n == null ? 1 : n + 1);
20            }
21        }
22        List<Entry<String, Integer>> s = new ArrayList<Entry<String, Integer>>(urls.entrySet());
23        Collections.sort(s, new Comparator<Entry<String, Integer>>() {
24            public int compare(Entry<String, Integer> o1,
25                               Entry<String, Integer> o2) {
26                return o2.getValue() - o1.getValue();
27            }
28        });
29
30        for(int i = 0; i != 10; ++i)
31            System.out.println(s.get(i).getKey() + " " + s.get(i).getValue());
32    }
33 }
```

Assinalar a verbosidade excessiva da implementação java



Implementação Java

```
1 import java.io.*;
2 import java.util.*;
3 import java.util.Map.Entry;
4 import java.util.regex.*;
5
6 public class WfReader {
7     public static void main(String[] args) throws IOException {
8         BufferedReader br = new BufferedReader(new FileReader(args[0]));
9         Map<String, Integer> urls = new HashMap<String, Integer>();
10        String linha;
11        Pattern p = Pattern.compile("GET /ongoing/When/\\d(3)x/\\d(4)/\\d\\d/\\d\\d/[^ .]+ ");
12        Matcher m = p.matcher("");
13
14        while((linha = br.readLine()) != null) {
15            m.reset(linha);
16            if(m.find()) {
17                String url = m.group(1);
18                Integer n = urls.get(url);
19                urls.put(url, n == null ? 1 : n + 1);
20            }
21        }
22        List<Entry<String, Integer>> s = new ArrayList<Entry<String, Integer>>(urls.entrySet());
23        Collections.sort(s, new Comparator<Entry<String, Integer>>() {
24            public int compare(Entry<String, Integer> o1,
25                               Entry<String, Integer> o2) {
26                return o2.getValue() - o1.getValue();
27            }
28        });
29
30        for(int i = 0; i != s.size(); ++i)
31            System.out.println(s.get(i).getKey() + " " + s.get(i).getValue());
32    }
33 }
```

O que está a mais



Implementação Groovy

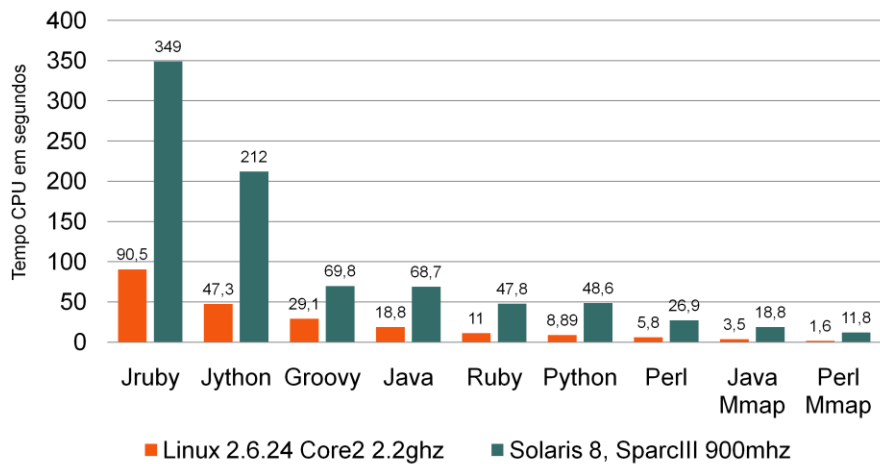
```
1 urls = [:]
2 m = "" =~ 'GET /ongoing/When/\\d{3}x/(\\d{4}/\\d\\d/\\d\\d/[^ .]+) '
3
4 new File(args[0]).eachLine {
5     if(m.reset(it).find()) {
6         total = urls[m.group(1)]
7         urls[m.group(1)] = total == null ? 1 : ++total
8     }
9 }
10
11 urls.entrySet().sort{-it.value}[0..10].each{
12     println it
13 }
```

Realçar: notação específica para maps, expressões regulares, iteração, closures

Voltar a falar do fosso semântico



Então e o desempenho?



Realçar :

Testes feitos com cache quente! Valores correspondem à média de 3 execuções do benchmark seguidas.

Jruby e Jython correspondem ao mesmo programa ruby e python, mas correndo sobre as implementações Jruby e Jython.

Java corresponde à implementação readline

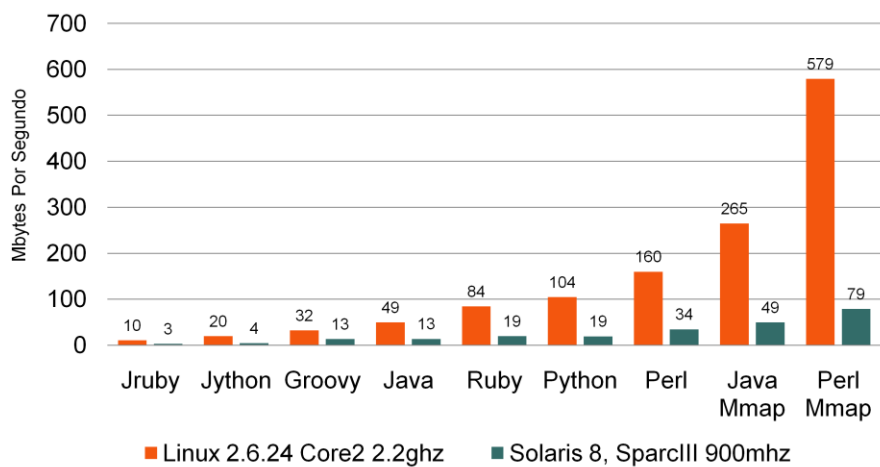
Todas as implementações processam linha a linha à exceção das Mmap, que processam em streaming

Java Mmap corresponde à utilização de um MappedByteBuffer (implementação java de memory mapped files)

- JRuby 1.1.1 e Ruby 1.8.6
- Jython 2.2.3 e Python 2.5.2
- Groovy 1.5.6
- A VM 1.6 foi usada em todos os testes



Desempenho de outro ponto de vista

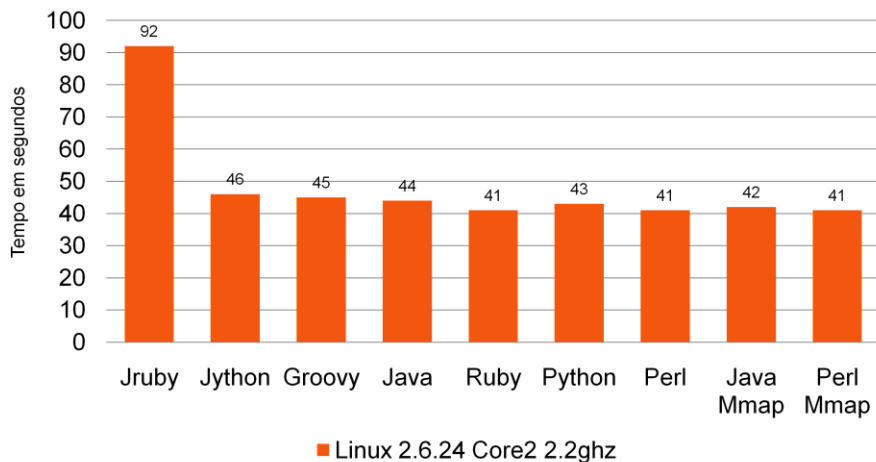


A maior parte das implementações consegue processar mais informação que o débito de I/O disponível



Desempenho no “Mundo Real” – cache fria

echo 1 > /proc/sys/vm/drop_caches



Dismistificar a “falta de performance” das linguagens de scripting

Realçar que na vida real o tempo de cpu não tem tanto impacto como isso porque a maior parte das aplicações são limitadas a nível de I/O e não de CPU.

Perguntar à audiência se alguém sabe como fazer reset da cache de leitura em solaris ou em windows.



Problemas com o scripting

- Fraca integração com ambientes de desenvolvimento
- Ainda são um alvo em movimento
- Características dinâmicas dificultam o refactoring e a análise estática
- Pool de recursos pequena
- Desempenho
- Os ganhos de produtividade diminuem à medida que a complexidade dos problemas a resolver aumenta

Nem tudo são rosas.

Realçar que à medida que os projectos se tornam complexos as vantagens do “scripting” diluem-se e a partir de certo ponto podem tornar-se prejudiciais.

Falar porque é difícil fazer refactoring com linguagens dinâmicas



Aplicações Híbridas

- Misto de Java com Scripting
- Core das aplicações em Java
- Customização com scripting
- Classes/Interfaces Java podem ser extendidas/implementadas com Java
- Java -> Scripting ou Scripting -> Java

Exemplo: Grails

Fina camada groovy sobre Hibernate,
Spring, etc etc etc

Podemos ter o bolo e comê-lo. Passar a ideia de que as aplicações podem ter uma componente estável em Java e os componentes mais dinâmicos com scripting



Como fazer a Ponte entre os dois Mundos

- Usando as APIs disponibilizadas pela linguagem directamente (maior desempenho e mais funcionalidades)
- JSR 223 – Scripting API (Incluido no Java 6)
<http://www.jcp.org/en/jsr/detail?id=223>
- Apache Bean Scripting Framework
<http://jakarta.apache.org/bsf/index.html>



Desenvolvimento Rápido para a Web

- Grails (Groovy) e Rails (JRuby)
- Arquitetura MVC sem complexidade
- Não obriga à repetição de configurações para fazer as mesmas coisas espalhadas pelo código
- Persistência Embutida (Hibernate com GORM e Ruby com ActiveRecord)
- Live Demo?



Links para scripting

<http://scripting.dev.java.net/>

- BeanShell
<http://www.beanshell.org/>
- JavaScript
<http://www.mozilla.org/rhino>
- Pnuts
<http://pnuts.dev.java.net/>
- TCL
<http://tcljava.sourceforge.net/>
- Scala
<http://www.scala-lang.org/>
- Groovy
<http://groovy.codehaus.org/>
- Python
<http://jython.sourceforge.net/>
- Ruby
<http://jruby.sourceforge.net/>
- Scheme
<http://sisc-scheme.org/>
- PHP
<http://www.caucho.com/resin-3.0/quercus/>