Dynamický překlad
Prakticky!

@JaroslavTulach
Oracle Labs
Jaroslav Tulach

1. NetBeans Founder – 20 years ago
5. Oracle Labs: Graal/Truffle – 2015 - now
Safe Harbor Statement

The preceding and following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle’s products remains at the sole discretion of Oracle.
Java History from NetBeans Perspective

• Xelfi student project: Java 1.0.2
  • AWT, Applets, 1st Java IDE written in Java

• NetBeans 2.0: Java 1.1
  • Swing preview, JavaBeans, Classpath model

• NetBeans 2.X: Java 1.2
  • Support for javax.swing

• NetBeans 3.0
  • Acquired by Sun Microsystems
  • Open Sourced, May 2000
Make it Practical!

• Epilogue of the API book
  • University vs. software engineering
  • Amazed by my first Linz visit

• Make a useful contribution!
  • No more: create and throw away
  • Tons of GitHub repositories (Truffle/Graal related)
  • Create a PR and make them accept it
  • Summer internship offer

http://practical.apidesign.org
Program Agenda

1. Graal VM overview
2. Speed up your language
3. Talk to your compiler
4. Ahead of time compilation
5. Tools – debugger, profiler & co.
6. Static languages on JVM
Aren’t you Already a Polyglot?
OracleLabs Vision

• Vision without speculation would be today's reality
• Radical innovation
• New approach to existing technologies
• Research + Engineering
• Hardware & Software
• Engineered to work together
• Java/bytecode/IR/assembly/processors/memory
Demo

Download GraalVM.org
Systems come with Various Interfaces
You can execute any language on the JVM / CLR - as long as it looks like Java / C#.
Can we Connect the Polyglots... Easily!?
Demo

Sieve of Eratosthenes with many Polyglot Flavors
GraalVM: One VM to Rule them all!
Graal Vision

High performance for all languages

Zero overhead interoperability between languages

Shared infrastructure and tooling across languages
Graal Virtual Machine

- Modern alternative to HotSpot C2
  - Maintainable code base
  - Toolable, approachable
  - Ready for today's code
  - JEP 243: Java Compiler Interface
- Partial evaluation
- Aggressive speculations
- Smooth de-optimizations
Demo
Understanding Your Compiler
### Current situation

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prototype a new language</td>
<td>Parser and language work to build syntax tree (AST), AST Interpreter</td>
</tr>
<tr>
<td>Write a &quot;real&quot; VM</td>
<td>In C/C++, still using AST interpreter, spend a lot of time implementing runtime system, GC, …</td>
</tr>
<tr>
<td>People start using it</td>
<td></td>
</tr>
<tr>
<td>People complain about performance</td>
<td>Define a bytecode format and write bytecode interpreter</td>
</tr>
<tr>
<td>Performance is still bad</td>
<td>Write a JIT compiler Improve the garbage collector</td>
</tr>
</tbody>
</table>

### How it should be

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prototype a new language in Java</td>
<td>Parser and language work to build syntax tree (AST) Execute using AST interpreter</td>
</tr>
<tr>
<td>Write a &quot;real&quot; VM</td>
<td>Integrate with VM-building framework</td>
</tr>
<tr>
<td>Integrate with Modular VM</td>
<td>Add small language-specific parts</td>
</tr>
<tr>
<td>People start using it</td>
<td>And it is already fast</td>
</tr>
</tbody>
</table>

---

*Truffle: Write your own language!*
Demo

Debugger, profiler, coverage & other tooling for free
Do you care about code or data?

Are C, C++, Java domain specific languages?

Static typing
Faster execution
Targeting larger projects that care about code

More flexible
Easier to use
Targeting exploratory programming, rapid prototyping, scripting

Java, C#
Visual Basic
Python, Ruby
JavaScript
R

JavaScript

Do you care about code or data?
One VM for all languages means interoperability and being able to choose the best language for the task!

The goal: The Power of Interop with Full Speed!
GraalVM Architecture

Graal Compiler

JVM Compiler Interface (JVMCI) JEP 243

Java HotSpot VM

Truffle Framework

Sulong (LLVM)

Ruby

R

JS

C

Java

Scala

Groovy
Fast Startup + Low Overhead

• Dynamic compilation
  • Speculation needs metadata to deoptimize

• Ahead of Time Compilation: SubstrateVM
  • Compile bytecode (Java, Kotlin, Scala) to native code
  • Native speed

• Java goes native
  • Compile Java libraries to native ones
  • Embed into existing runtimes
Demo
Ahead-of-time compilation
The Biggest Problem of Dynamic Languages

\[ a + b \]

- int
- String
- Object
How Does it Work?
In case of Invalid Assumptions...
Become Polyglot!

• Try it now!
  – Download JVM:  http://graalvm.org
  – Open source:  http://github.com/oracle/graal

• Mix it all!
  – JRuby, Python, JavaScript, R, C, node.js

• Speed your own language up!
  – Propose a project - nswi176@d3s.mff.cuni.cz

• We are hiring!  Praha, Zurich, Linz