JRuby+Truffle
A tour through a new Ruby implementation

Chris Seaton
@ChrisGSeaton
Oracle Labs

Benoit Daloze
@eregontp
JKU Linz

Kevin Menard
@nirvdrum
Oracle Labs

Ruby logo copyright (c) 2006, Yukihiro Matsumoto, licensed under the terms of the Creative Commons Attribution-ShareAlike 2.5 agreement
Safe Harbor Statement

The following is intended to provide some insight into a line of research in Oracle Labs. 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. Oracle reserves the right to alter its development plans and practices at any time, and the development, release, and timing of any features or functionality described in connection with any Oracle product or service remains at the sole discretion of Oracle. Any views expressed in this presentation are my own and do not necessarily reflect the views of Oracle.
What is the big idea?
Current situation

- Prototype a new language
  - Parser and language work to build syntax tree (AST), AST Interpreter
- Write a “real” VM
  - In C/C++, still using AST interpreter, spend a lot of time implementing runtime system, GC, ...
- People start using it
- People complain about performance
  - Define a bytecode format and write bytecode interpreter
- Performance is still bad
  - Write a JIT compiler
  - Improve the garbage collector
Prototype a new language
Parser and language work to build syntax tree (AST), AST Interpreter
Write a “real” VM
In C/C++, still using AST interpreter, spend a lot of time implementing runtime system, GC, ...
People start using it
People complain about performance
Define a bytecode format and write bytecode interpreter
Performance is still bad
Write a JIT compiler
Improve the garbage collector

Prototype a new language in Java
Parser and language work to build syntax tree (AST)
Execute using AST interpreter
People start using it
And it is already fast

Current situation  How it should be
What are Truffle and Graal?
Truffle: a framework for writing AST interpreters for languages in Java

Graal: a dynamic compiler (JIT) for Java, written in Java, as a Java library
Node Rewriting for Profiling Feedback

Node Transitions

AST Interpreter Uninitialized Nodes

AST Interpreter Rewritten Nodes

Compilation using Partial Evaluation

Compiled Code

codon.com/compilers-for-free
Frequently executed call
Deoptimization to AST Interpreter

Node Rewriting to Update Profiling Feedback

Recompilation using Partial Evaluation

t1 = Fixnum(a) + Fixnum(b)
if t1.overflowed?
    t1 = Bignum(a) + Bignum(b)
    t2 = Bignum(t1) + Bignum(c)
else
    t2 = Fixnum(t1) + Fixnum(c)
    if t2.overflowed?
        t2 = Bignum(t1) + Bignum(c)
    end
end
end
t1 = Fixnum(a) + Fixnum(b)
deooptimize! if t1.overflowed?
t2 = Fixnum(t1) + Fixnum(c)
deooptimize! if t2.overflowed?
Guest Language

Bytecode

JVM
Guest Language

Java IR, machine code cache, invalidation and deoptimisation, optimisation phases, replacements, etc... etc...

Graal VM
Guest Language

AST interpreter

Truffle

Graal VM
A tour through Ruby, Truffle and Graal
Specializations

class Array

  def [](index=Fixnum())
    # return element at index
  end

  def [](index=Fixnum(), num=Fixnum())
    # return num elements starting at index
  end

  def [](range=Range())
    # return elements from range.start to range.end
  end

  def [](index)
    # coerce index and dispatch
  end

Specializations

def clamp(num, min, max)
    [min, num, max].sort[1]
end

chunky_png and psd.rb, Willem van Bergen, Ryan LeFevre, Kelly Sutton, Layer Vault, Floorplanner et al
From Fixnum#+ to 0x03 0x70
Digging through ObjectSpace and deoptimization

- Deoptimize
- Get a consistent view of memory: safepoints
- Find all reachable objects
- Iterate through them
public Map<Long, RubyBasicObject> collectLiveObjects() {
    liveObjects = new HashMap<>();

    visitor = new ObjectGraphVisitor() {
        @Override
        public boolean visit(RubyBasicObject object) {
            return liveObjects.put(object.getObjectID(), object) == null;
        }
    };

    context.getSafepointManager().pauseAllThreadsAndExecute(new Consumer<RubyThread>() {
        @Override
        public void accept(RubyThread currentThread) {
            synchronized (liveObjects) {
                visitor.visit(currentThread);
                context.getCoreLibrary().getGlobalVariablesObject().visitObjectGraph(visitor);
                context.getCoreLibrary().getMainObject().visitObjectGraph(visitor);
                context.getCoreLibrary().getObjectClass().visitObjectGraph(visitor);
                visitCallStack(visitor);
            }
        }
    });

    return Collections.unmodifiableMap(liveObjects);
}
Does it really implement Ruby?
93% RubySpec language specs
Brian Shirai et al
24%  RubySpec core library specs
Brian Shirai et al
How fast is it?
Mandelbrot

Speedup Relative to MRI 2.1.2

<table>
<thead>
<tr>
<th>Language</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1.2</td>
<td></td>
</tr>
<tr>
<td>Rbx 2.2.26</td>
<td></td>
</tr>
<tr>
<td>JRuby 1.7.12</td>
<td></td>
</tr>
<tr>
<td>Topaz</td>
<td></td>
</tr>
<tr>
<td>JRuby+Truffle</td>
<td></td>
</tr>
<tr>
<td>V8</td>
<td></td>
</tr>
<tr>
<td>Java</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
</tr>
</tbody>
</table>
chunky_png and psd.rb

Willem van Bergen, Ryan LeFevre, Kelly Sutton, Layer Vault, Floorplanner et al

Ruby logo copyright (c) 2006, Yukihiro Matsumoto, licensed under the terms of the Creative Commons Attribution-ShareAlike 2.5 agreement
github.com/jruby/bench9000
How will we solve startup time, memory footprint and the JVM dependency?
Ahead-of-Time Compilation

Static Analysis

Java Application
JDK
Substrate VM

All Java classes from application, JDK, and Substrate VM

Reachable methods, fields, and classes

Initial Heap
Machine Code
OS

Application running without compilation or class loading
<table>
<thead>
<tr>
<th>MRI</th>
<th>JRuby</th>
<th>Truffle on JVM</th>
<th>Truffle on SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>353</td>
<td>688</td>
<td>14</td>
</tr>
</tbody>
</table>

Execution Time

<table>
<thead>
<tr>
<th>MRI</th>
<th>JRuby</th>
<th>Truffle on JVM</th>
<th>Truffle on SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>35</td>
<td>53</td>
<td>9</td>
</tr>
</tbody>
</table>

Memory Footprint
How do we implement C extensions?
MRI With C Extension
Rubinius With C Extension
JRuby With C Extension
JRuby+ Truffle With C Extension

Mean Speedup Relative to MRI Without C Extension (s/s)

0.0 5.0 10.0 15.0 20.0 25.0 30.0 35.0

2.5 4.2 10.6 32.0
How does it build on other projects?
• Lexer, parser
• Strings, regexps, IO
• Command line
• Build and distribution infrastructure
• Cannot re-use more of the core library due to very different approaches

• The language design
• The parts of the standard library written in Ruby
• Considering trying to use some of the C code

• Parts of the core library
• Parts of the standard library
• RubySpec
• We have our own implementations of the Rubinius primitives

JRuby logo copyright (c) Tony Price 2011, licensed under the terms of the Creative Commons Attribution-NoDerivs 3.0 Unported (CC BY-ND 3.0)
Ruby logo copyright (c) 2006, Yukihiro Matsumoto, licensed under the terms of the Creative Commons Attribution-ShareAlike 2.5 agreement
Rubinius logo licensed under the terms of the Creative Commons Attribution-NoDerivs 3.0 Unported
What other big ideas do we have?
Wrapping up


C. Seaton, M. L. Van De Vanter, and M. Haupt. *Debugging at full speed*. In Proceedings of the 8th Workshop on Dynamic Languages and Applications (DYLA), 2014


#jruby

github.com/jruby/jruby

chrissieaton.com/rubytruffle

@chrisgseaton   @eregontp   @nirvdrum
Safe Harbor Statement

The preceding is intended to provide some insight into a line of research in Oracle Labs. 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. Oracle reserves the right to alter its development plans and practices at any time, and the development, release, and timing of any features or functionality described in connection with any Oracle product or service remains at the sole discretion of Oracle. Any views expressed in this presentation are my own and do not necessarily reflect the views of Oracle.
Hardware and Software
Engineered to Work Together