
ORACLE®

Implementing Ruby Using Truffle and Graal

Chris Seaton @ChrisGSeaton

ECOOP Summer Schools
2014

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.

We're going to talk about

1. Motivation
2. Truffle and Graal Theory
3. Truffle and Graal in Practice
4. Applying it to Ruby

Motivation

JavaScript: **One language to rule them all** | VentureBeat

venturebeat.com/2011/.../javascript-**one-language-to-rule-them**-...

by Peter Yared - in 23 Google+ circles

Jul 29, 2011 - Why code in two different scripting languages, one on the client and one on the server? It's time for **one language to rule them all**. Peter Yared **...**

[PDF] Python: **One** Script (**Language**) **to rule them all** - Ian Darwin

www.darwinsys.com/python/python4unix.pdf

Another **Language**? ▸ Python was invented in 1991 by Guido van. Rossum. ▫ Named after the comedy troupe, not the snake. ▸ Simple. ▫ They **all** say that!

Q & Stuff: **One Language to Rule Them All - Java**

qstuff.**blogspot**.com/2005/10/**one-language-to-rule-them-all-java**.html

Oct 10, 2005 - **One Language to Rule Them All - Java**. For a long time I'd been hoping to add a scripting language to LibQ, to use in any of my (or other **...**

Dart : **one language to rule them all** - MixIT 2013 - Slideshare

fr.slideshare.net/sdeleuze/dart-mixit2013en

DartSébastien Deleuze - @sdeleuzeMix-IT 2013One **language to rule them all** ...

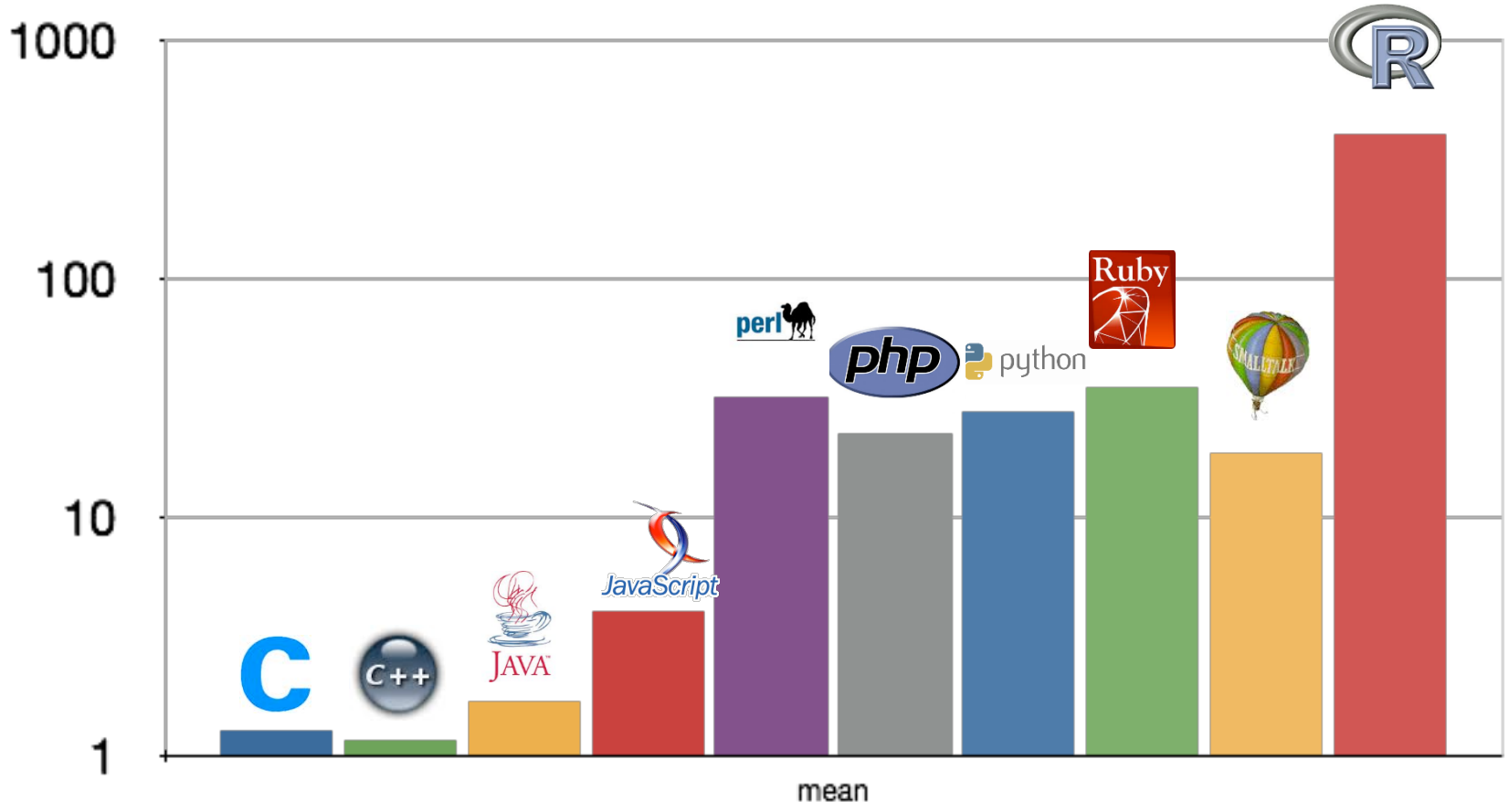
Questions | Tags | Tour | Users

Stack Overflow is a question and answer site for professional and enthusiast programmers. It's 100% free, no registration required.

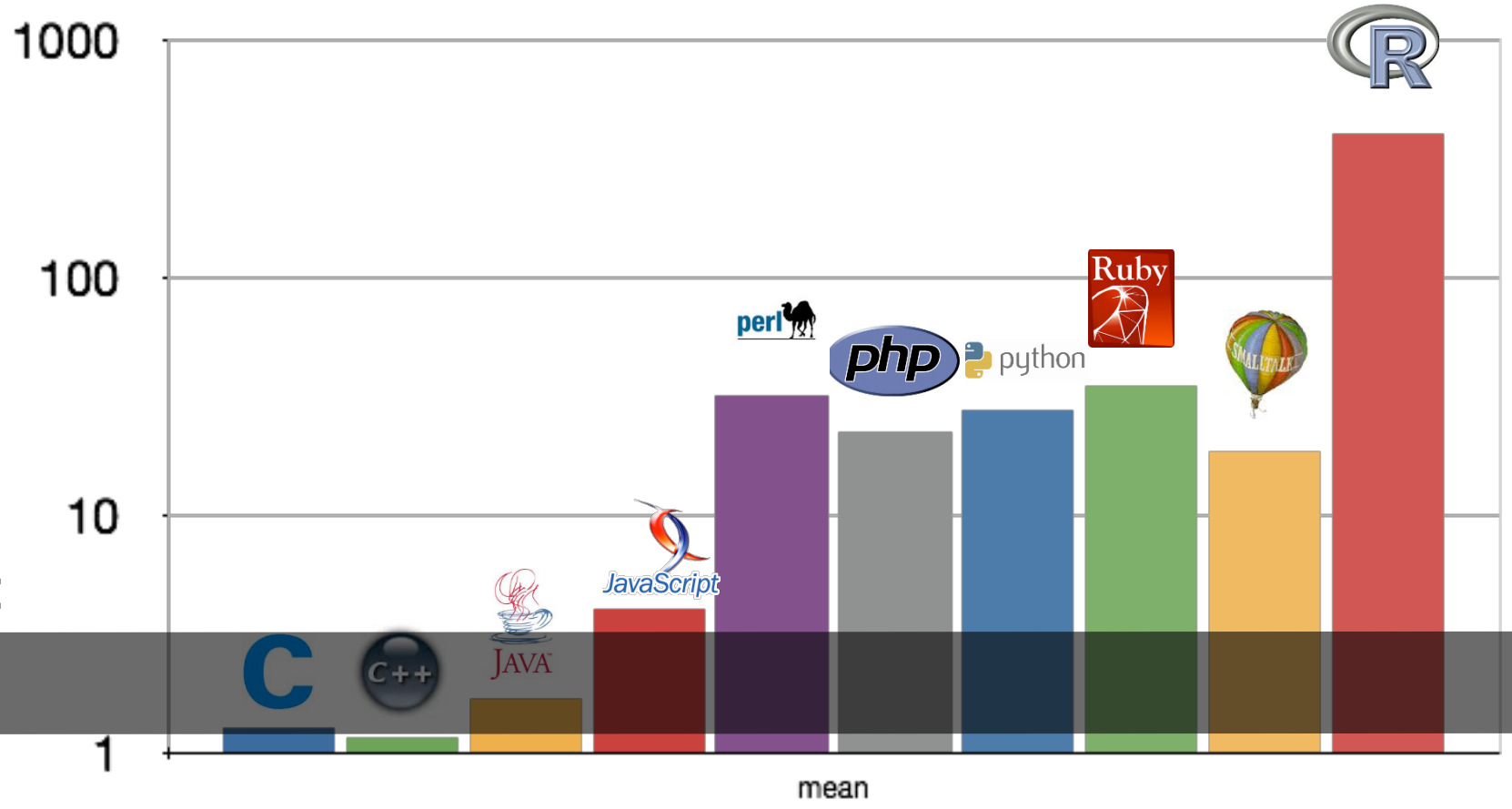
Why can't there be an “ultimate” programming language?

closed as not constructive by Tim, Bo Persson, Devon_C_Miller, Mark, Graviton Jan 17 at 5:58

Computer Language Benchmarks Game



Computer Language Benchmarks Game



Goal:

Current situation

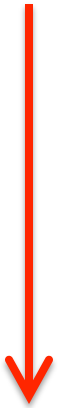
- Prototype a new language
  - Parser and language work to build syntax tree (AST), AST Interpreter
- Write a “real” VM
  - In C/C++, still using AST interpreter, spend a lot of time implementing runtime system, GC, …
- People start using it
- People complain about performance
  - Define a bytecode format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

How it should be

- 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

Truffle and Graal Theory

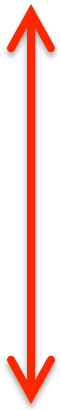
Guest Language



Bytecode

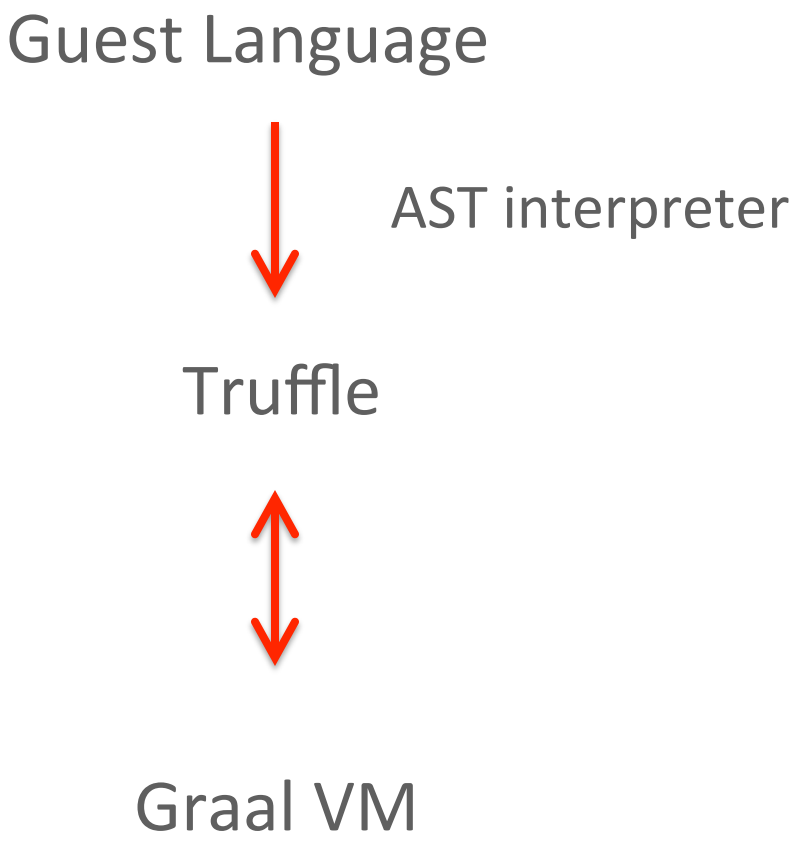
JVM

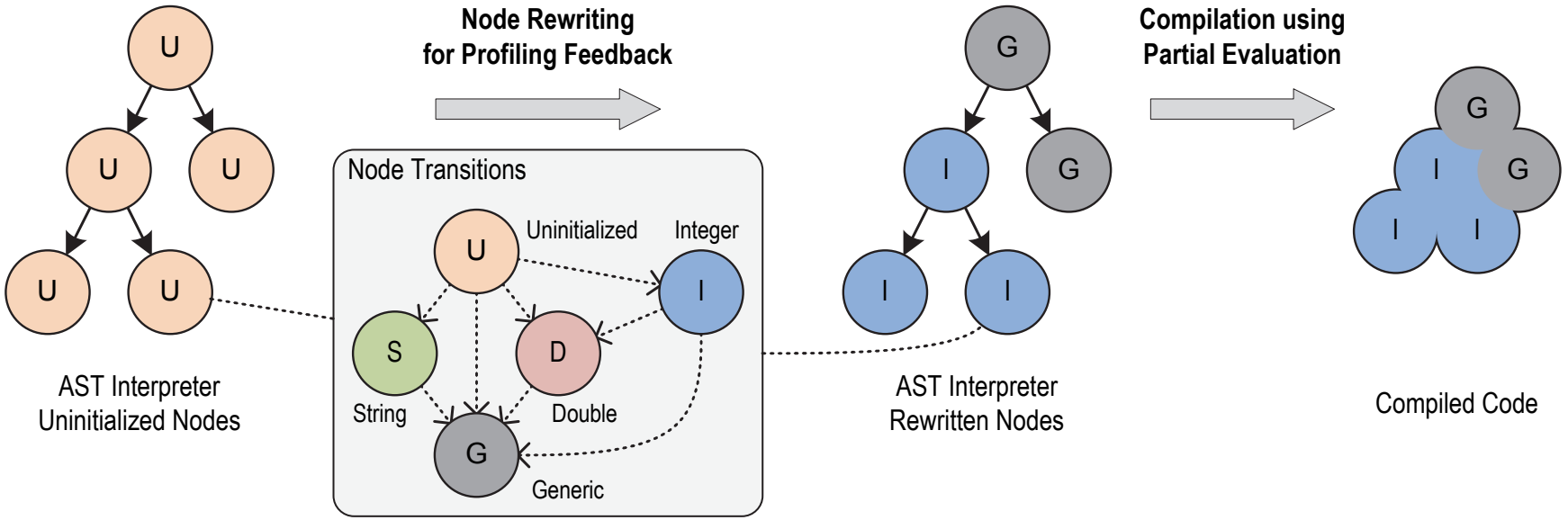
Guest Language



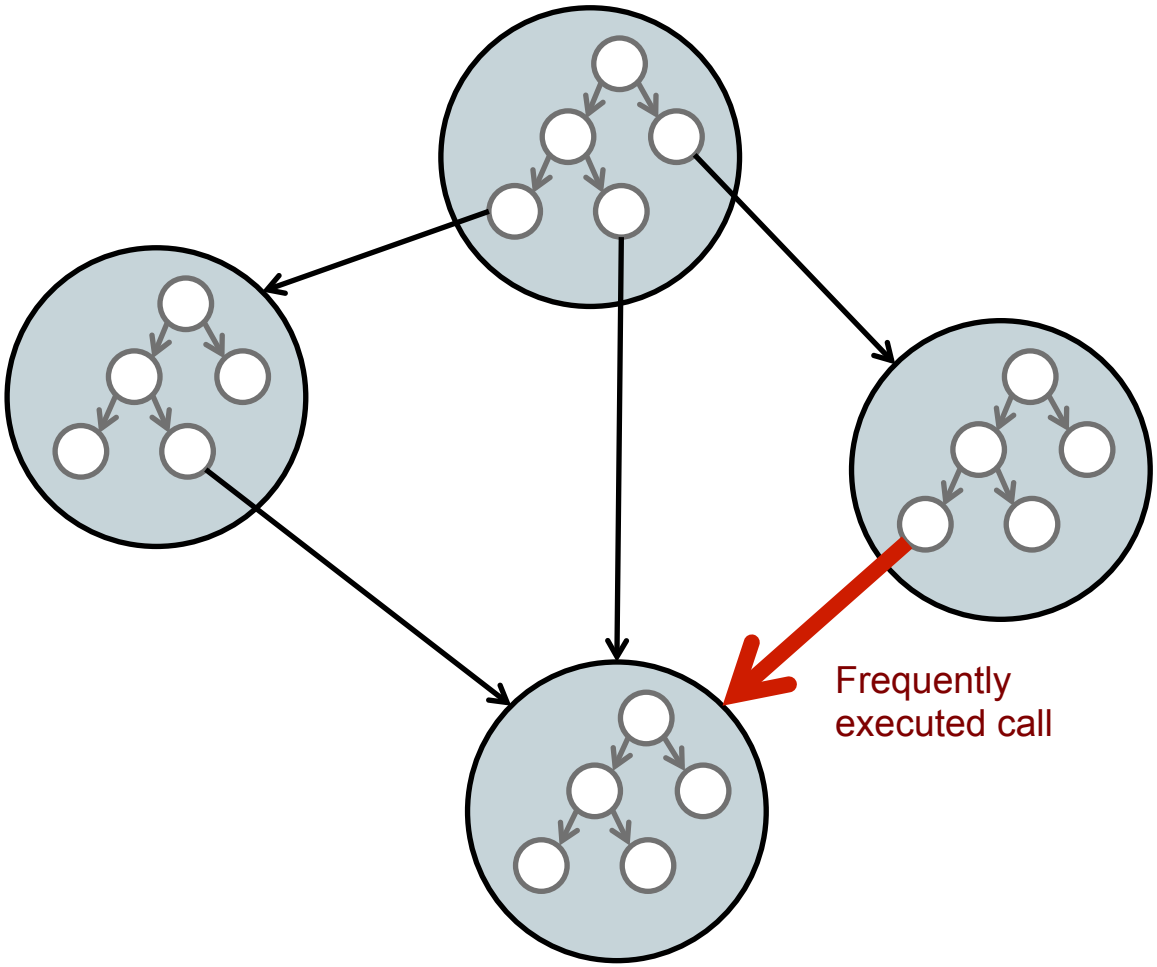
Java IR, machine code cache, invalidation and deoptimisation, optimisation phases, replacements, etc… etc…

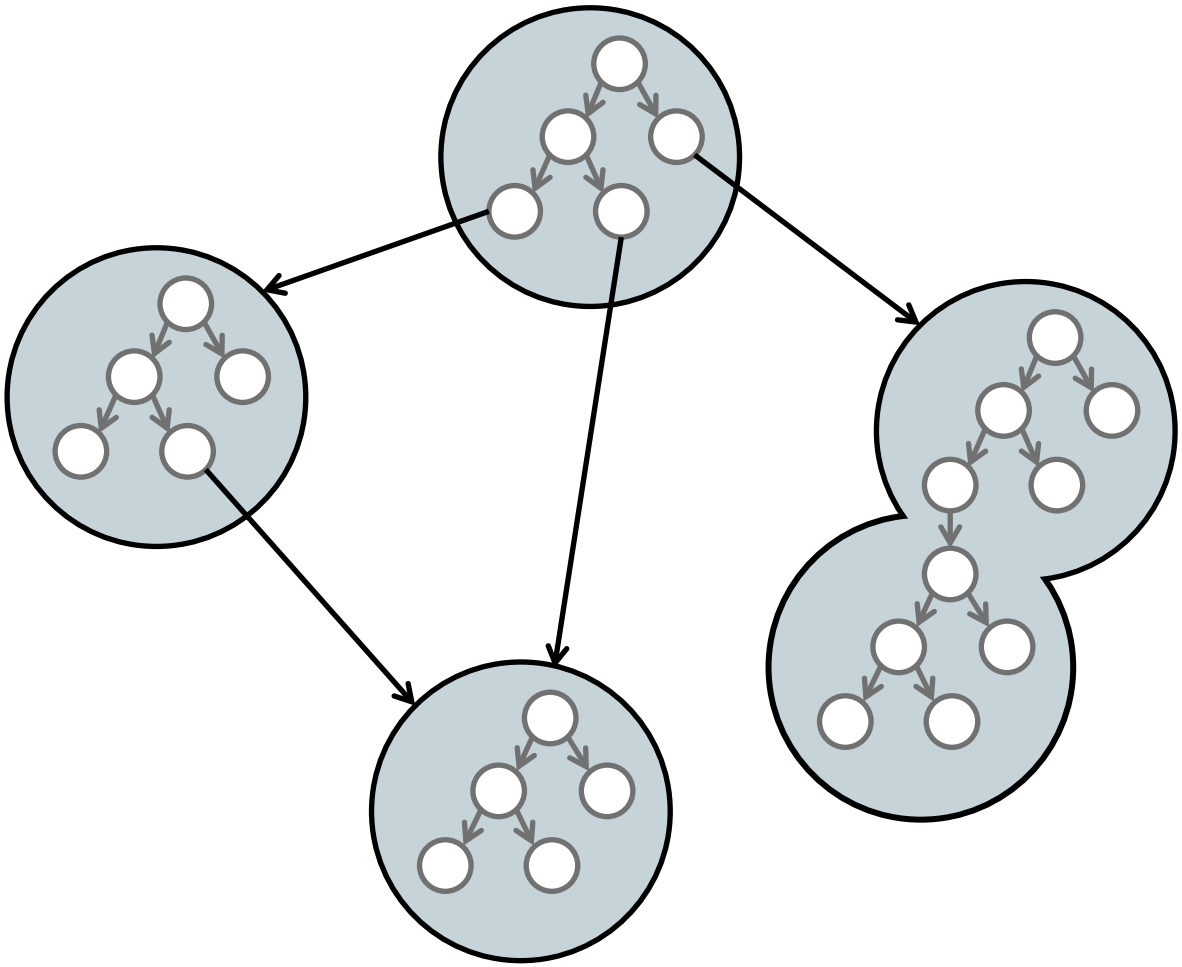
Graal VM

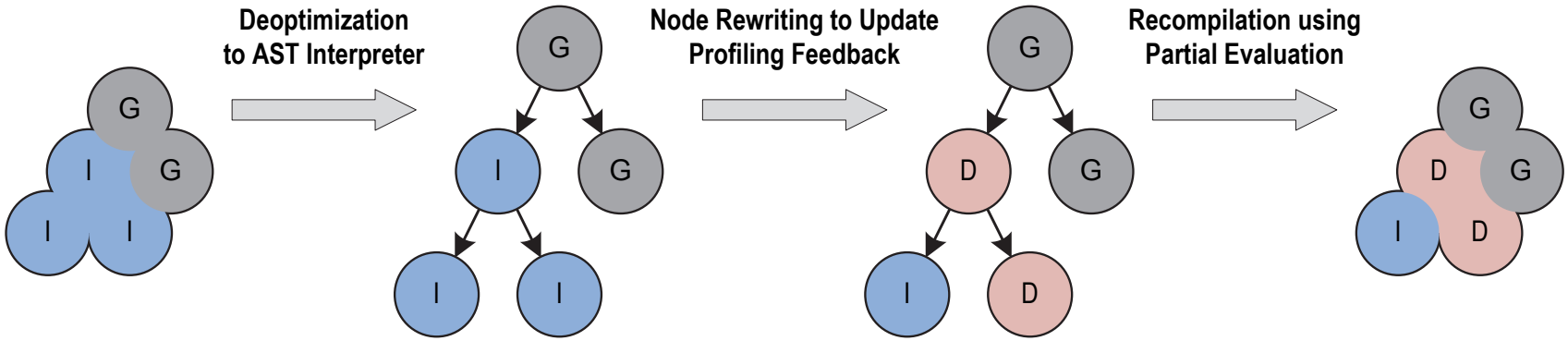




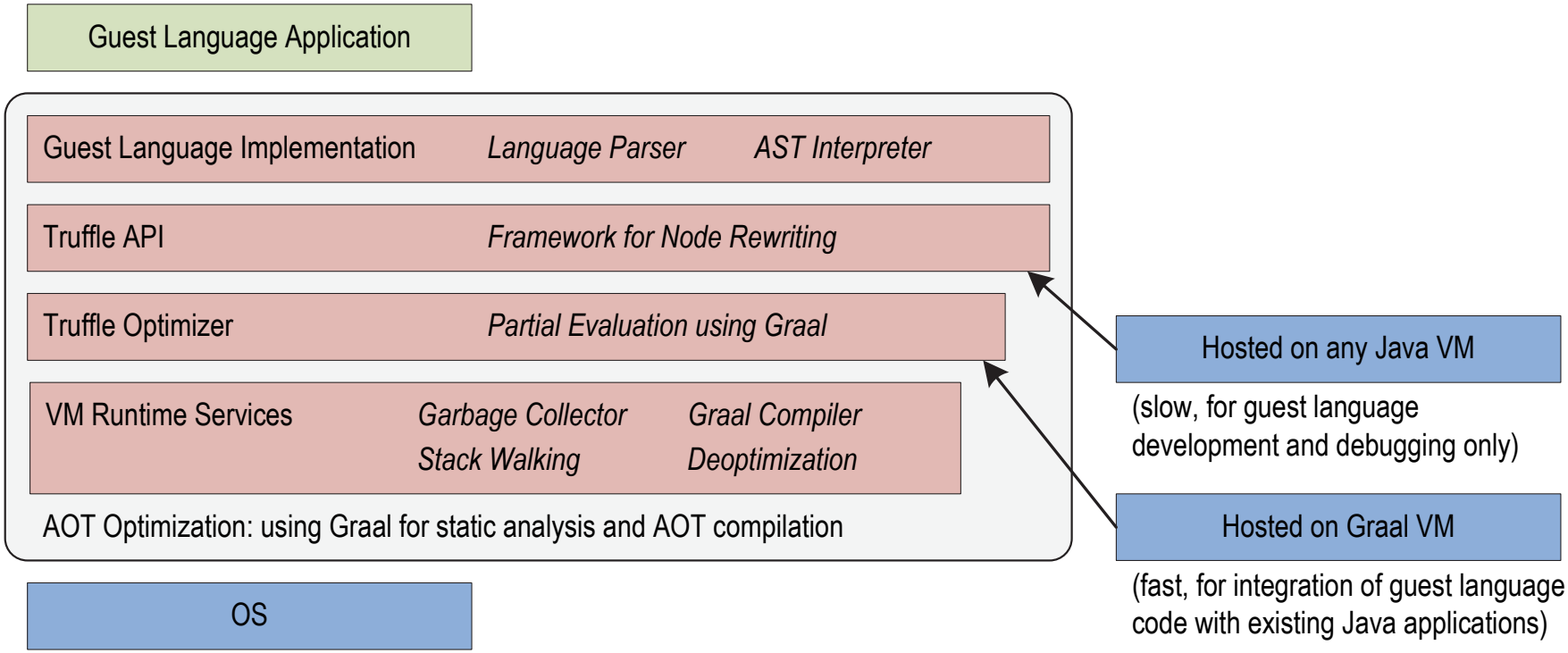
T. Würthinger, C. Wimmer, A. Wöß, L. Stadler, G. Duboscq, C. Humer, G. Richards, D. Simon, and M. Wolczko. One VM to rule them all. In Proceedings of Onward!, 2013.







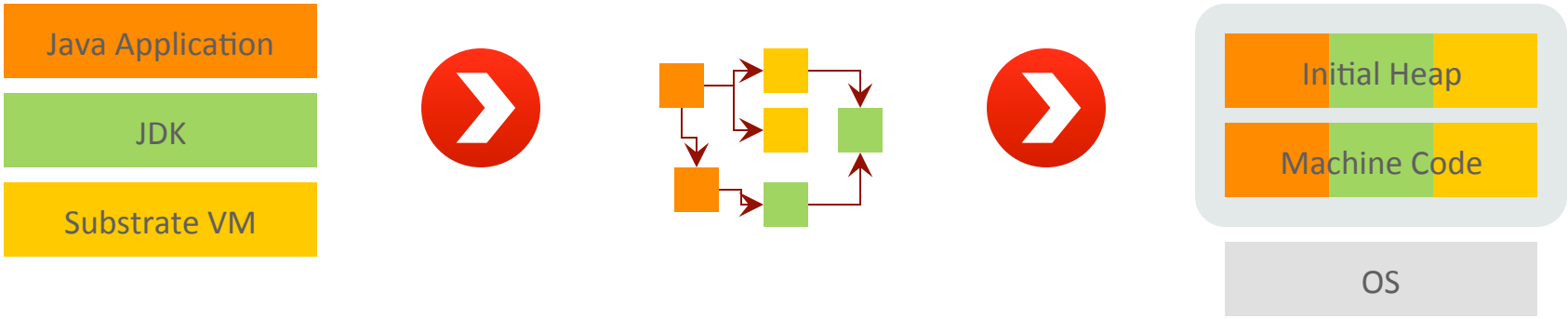
T. Würthinger, C. Wimmer, A. Wöß, L. Stadler, G. Duboscq, C. Humer, G. Richards, D. Simon, and M. Wolczko. One VM to rule them all. In Proceedings of Onward!, 2013.



T. Würthinger, C. Wimmer, A. Wöß, L. Stadler, G. Duboscq, C. Humer, G. Richards, D. Simon, and M. Wolczko. One VM to rule them all. In Proceedings of Onward!, 2013.

Static Analysis

Ahead-of-Time Compilation



All Java classes from application, JDK, and Substrate VM

Reachable methods, fields, and classes

Application running without compilation or class loading

One VM

- Good interpreted performance on a standard JVM
- Extremely good dynamically compiled performance on Graal
- High level representation of languages
- Substrate VM for startup performance, low footprint and easy distribution

- JavaScript, **Ruby**, R, J, C, Python, SmallTalk

Truffle and Graal in Practice

Simple Language (SL)

- Minimal language for demonstration and documentation
- Similar to JavaScript
- Included in the OpenJDK Graal repository

Setup

- `hg clone http://hg.openjdk.java.net/graal/graal`
- `cd graal`
- ./mx.sh --vm server build
- ./mx.sh ideinit

- Or just Google ‘graal openjdk’

```
public class SLIfNode extends SLStatementNode {
  @Child private SLExpressionNode conditionNode;
  @Child private SLStatementNode thenPartNode;
  @Child private SLStatementNode elsePartNode;

  public SLIfNode(SLExpressionNode conditionNode,
   SLStatementNode thenPartNode, SLStatementNode elsePartNode) {
    this.conditionNode = conditionNode;
    this.thenPartNode = thenPartNode;
    this.elsePartNode = elsePartNode;
  }

  public void executeVoid(VirtualFrame frame) {
    if (conditionNode.executeBoolean(frame)) {
      thenPartNode.executeVoid(frame);
    } else {
      elsePartNode.executeVoid(frame);
    }
  }
}
```

```
public class SLBlockNode extends SLStatementNode {
  @Children private final SLStatementNode[] bodyNodes;

  public SLBlockNode(SLStatementNode[] bodyNodes) {
    this.bodyNodes = adoptChildren(bodyNodes);
  }

  @ExplodeLoop
  public void executeVoid(VirtualFrame frame) {
    for (SLStatementNode statement : bodyNodes) {
      statement.executeVoid(frame);
    }
  }
}
```

```
public class SLReturnNode extends SLStatementNode {
  @Child private SLExpressionNode valueNode;
  ...
  public void executeVoid(VirtualFrame frame) {
    throw new SLReturnExceptn(valueNode.executeGeneric(frame));
  }
}
```

```
public class SLFunctionBodyNode extends SLExpressionNode {
  @Child private SLStatementNode bodyNode;
  ...
  public Object executeGeneric(VirtualFrame frame) {
    try {
      bodyNode.executeVoid(frame);
    } catch (SLReturnException ex) {
      return ex.getResult();
    }
    return SLNull.SINGLETON;
  }
}
```

```
public class SLReturnException
    extends ControlFlowException {
  private final Object result;
}
```

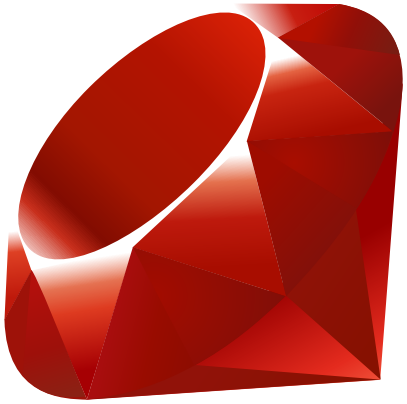
```
public class SLAddNode extends SLExpressionNode {
  @Child private SLExpressionNode leftNode;
  @Child private SLExpressionNode rightNode;

  @Override
  public Object executeGeneric(VirtualFrame frame) {
    Object left = leftNode.executeGeneric(frame);
    Object right = rightNode.executeGeneric(frame);

    if (left instanceof Long && right instanceof Long) {
      try {
        return ExactMath.addExact((Long) left, (Long) right);
      } catch (ArithmeticException ex) { }
    }

    if (left instanceof Long) {
      left = BigInteger.valueOf((Long) left);
    }
    if (right instanceof Long) {
      right = BigInteger.valueOf((Long) right);
    }
    if (left instanceof BigInteger && right instanceof BigInteger) {
      return ((BigInteger) left).add((BigInteger) right);
    }

    if (left instanceof String || right instanceof String) {
      return left.toString() + right.toString();
    }

    throw new UnsupportedSpecializationException(this, ...);
  }
}
```

```
  public Object executeGeneric(VirtualFrame frame) {
    Object left = leftNode.executeGeneric(frame);
    Object right = rightNode.executeGeneric(frame);

    if (left instanceof Long && right instanceof Long) {
      try {
        return ExactMath.addExact((Long) left, (Long) right);
      } catch (ArithmeticException ex) { }
    }

    if (left instanceof Long) {
      left = BigInteger.valueOf((Long) left);
    }
    if (right instanceof Long) {
      right = BigInteger.valueOf((Long) right);
    }
    if (left instanceof BigInteger && right instanceof BigInteger) {
      return ((BigInteger) left).add((BigInteger) right);
    }

    if (left instanceof String || right instanceof String) {
      return left.toString() + right.toString();
    }
...
```

```
@Specialization(rewriteOn = ArithmeticException.class)
protected long add(long left, long right) {
  return ExactMath.addExact(left, right);
}

@Specialization
protected BigInteger add(BigInteger left, BigInteger right) {
  return left.add(right);
}

@Specialization(guards = "isString")
protected String add(Object left, Object right) {
  return left.toString() + right.toString();
}

protected boolean isString(Object a, Object b) {
  return a instanceof String || b instanceof String;
}
```

Ruby in Truffle and Graal

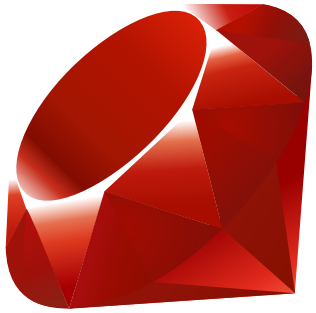
Introduction to Ruby



- Imperative, object oriented, dynamically typed
- Inspirations from Smalltalk and Perl
- Widely used with the Rails framework for web applications
- But also used in graphics, bioinformatics, systems, etc

Ruby Logo (Copyright (c) 2006, Yukihiro Matsumoto. Licensed under the terms of Creative Commons Attribution-ShareAlike 2.5.)

Ruby Implementations - MRI



- Implemented in C
- Bytecode interpreter
- Very simple optimisations – inline caches in instructions
- Probably the slowest commonly used interpreter there is

Ruby Implementations - Rubinius

- Implemented in C++ and Ruby
- Uses an LLVM-based JIT

Rubinius logo copyright 2011 Roger Bacardit. Attribution-NoDerivs 3.0 Unported (CC BY-ND 3.0)

Ruby Implementations - Topaz

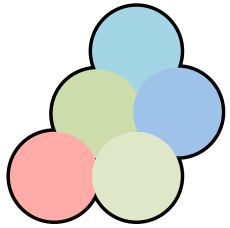
- Implemented in RPython
- Interpreter is statically compiled to native code via C
- Ruby code is compiled using a tracing JIT compiler

PyPy logo http://www.pypy.org/

Ruby Implementations - JRuby

- Implemented in Java
- Driver and primary user of JSR 292 (invokedynamic) until Nashorn
- AST interpreter -> bytecode compiler - > JIT by JVM
- Now looking at their own IR before bytecode

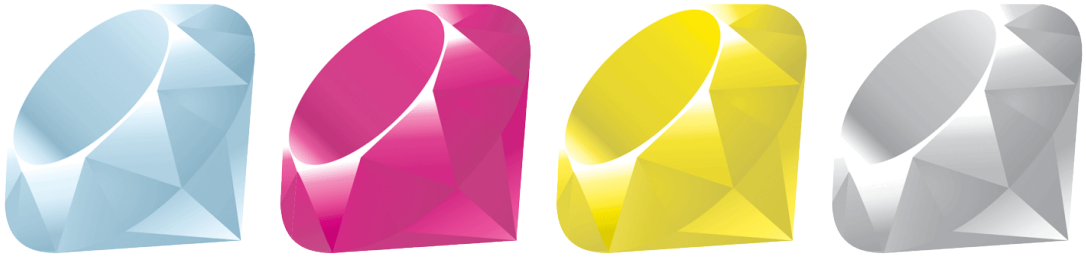
Ruby Implementations – JRuby+Truffle



Truffle

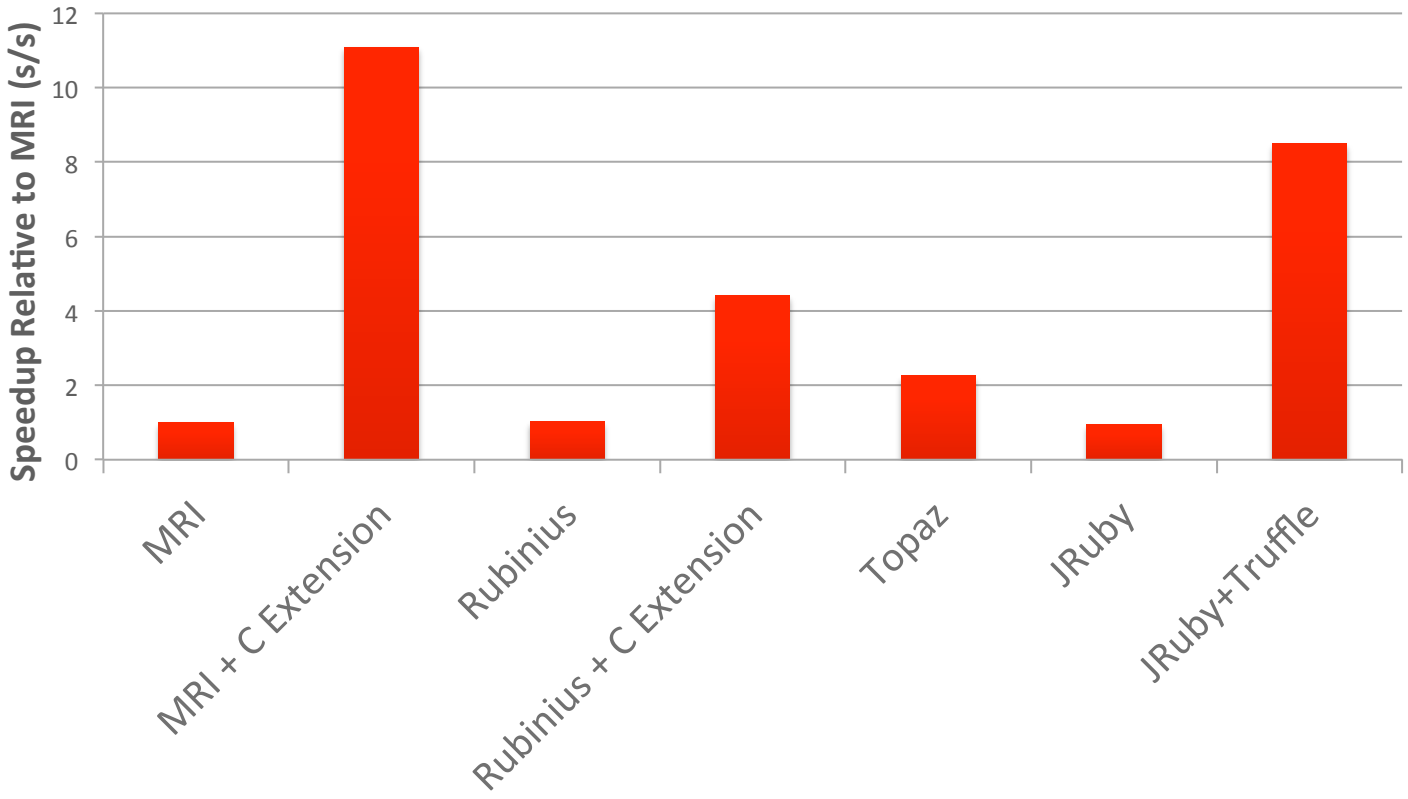
- Uses JRuby's parser and limited parts of their runtime
- Currently not much more than a tenant within JRuby
- AST interpreter, written using Truffle
- Works on a normal JVM
- Can implicitly use Graal VM

Benchmarks



- chunky_png and psd.rb
- Real code, unmodified from the original libraries
- Range of styles of Ruby code:
  - High performance tight numerical loops with local variables
  - Object oriented code such as method calls and instance variables
  - Ruby dynamic programming features such as #send

Performance on chunky_png and psd.rb



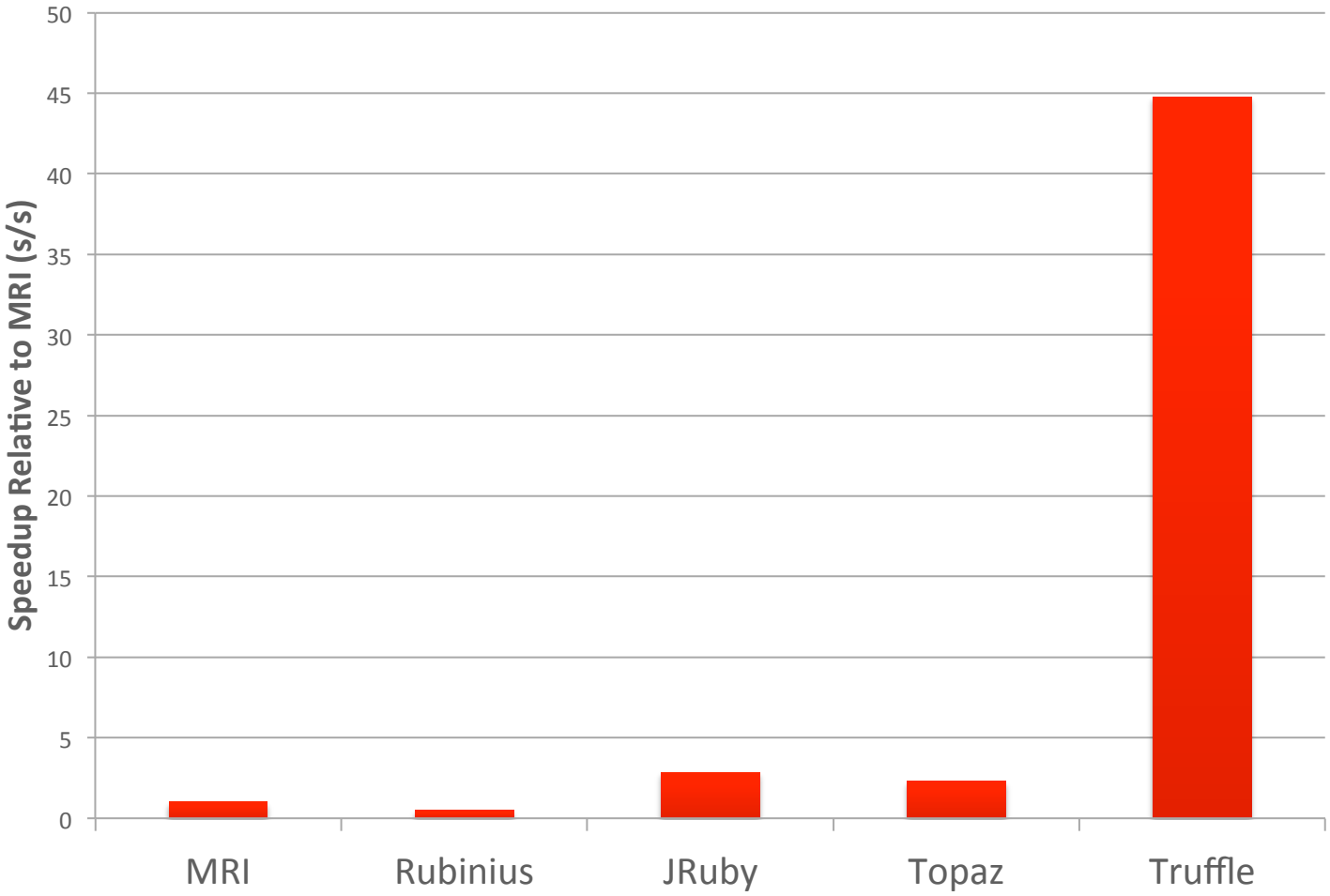
chrisseaton.com/rubytruffle/pushing-pixels

```
new_r = blend_channel(r(bg), r(fg), mix_alpha)

...

def method_missing(method, *args, &block)
  return ChunkyPNG::Color.send(method, *args) ↩
    if ChunkyPNG::Color.respond_to?(method)
  normal(*args)
end
```

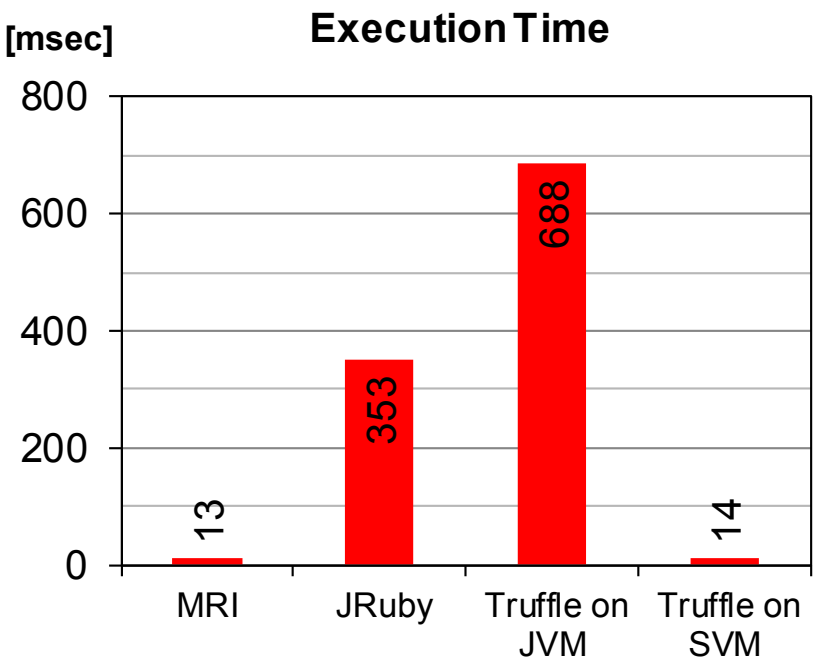
```
module Foo
  extend self

  def foo(a, b, c)
    hash = {a: a, b: b, c: c}
    array = hash.map { |k, v| v }
    x = array[0]
    y = [a, b, c].sort[1]
    x + y
  end

end
```

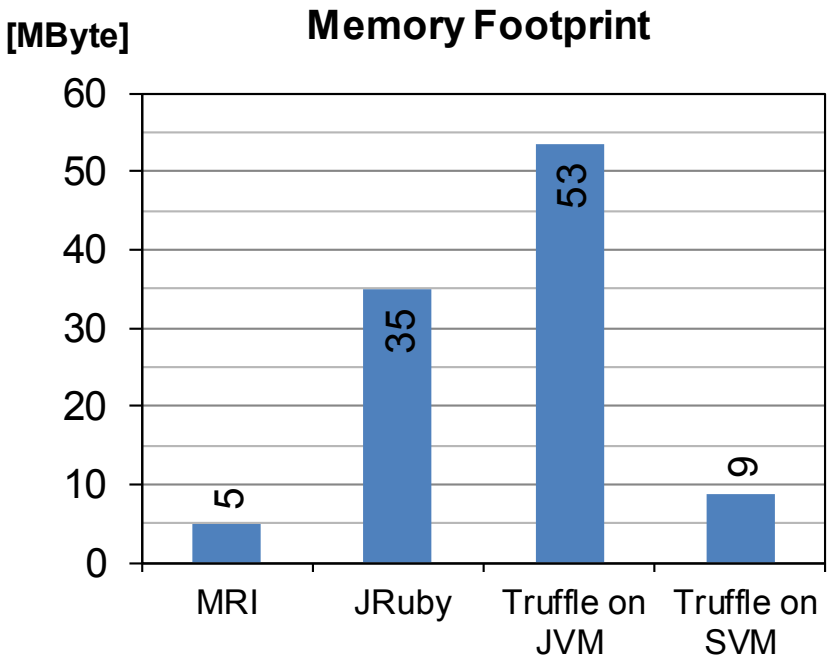
```
class Bar

  def method_missing(method, *args)
    if Foo.respond_to?(method)
      Foo.send(method, *args)
    else
      0
    end
  end

end
```

```
Bar.new.foo(14, 8, 6) => 22
```

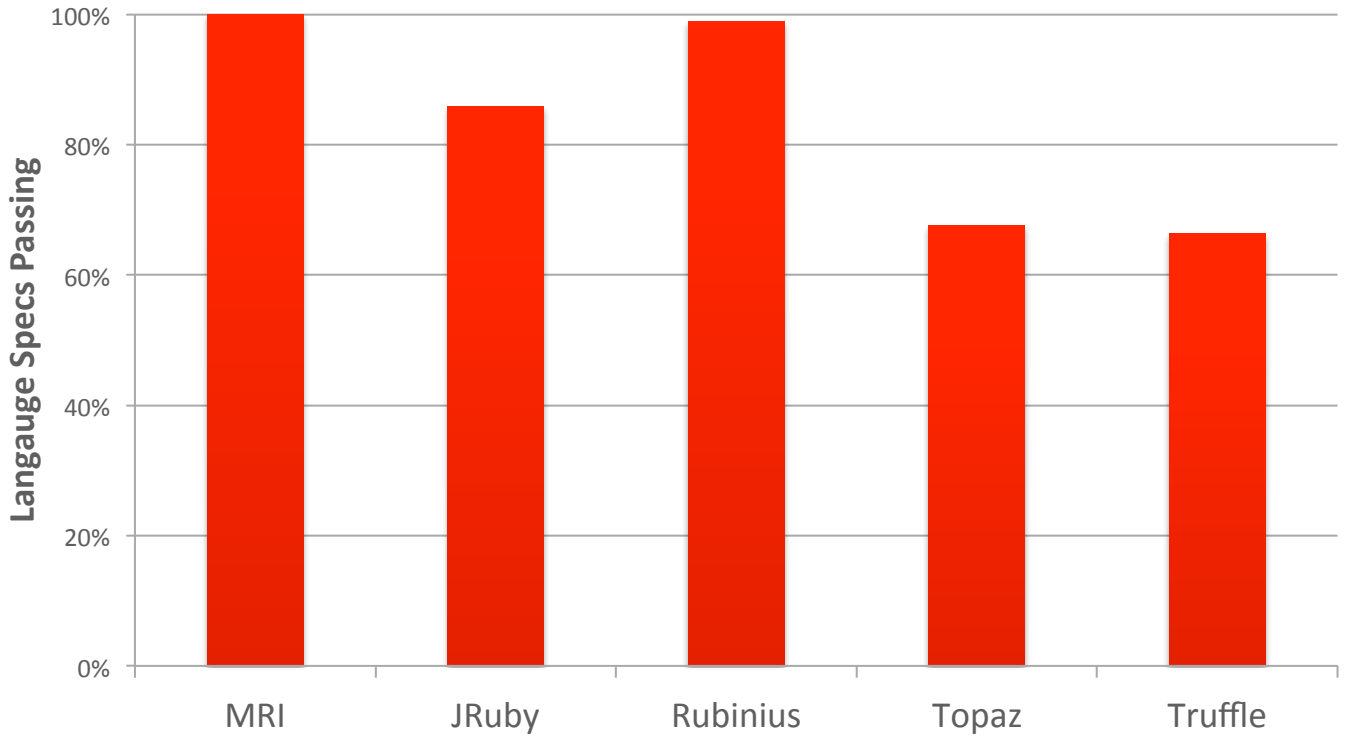


Hello World on Substrate VM





RubySpec



rubyspec.org – thanks to Brian Shirai et al

Language Feature	Implemented	Notes
Fixnum to Bignum promotion	✓	
Support for floating point	✓	
Closures	✓	
Bindings and eval	✓	Works from aliased methods
callcc and Continuation	✓	Limited support, the same as JRuby
Fibers	✓	Limited support, the same as JRuby
Frame local variables	✓	
C extensions	✓	Early work, but runs real C extensions
Ruby 1.9 encoding	✓	
Garbage collection	✓	
Concurrency and parallelism	✓	We currently use a GIL
Tracing and debugging	✓	Always enabled
ObjectSpace	✓	Always enabled
Method invalidation	✓	
Constant invalidation	✓	
Ruby on Rails		

Charles Nutter: ‘So You Want to Optimize Ruby’ http://blog.headius.com/2012/10/so-you-want-to-optimize-ruby.html

Language Feature	Implemented	Notes
Fixnum to Bignum promotion	✓	
Closures	✓	
Bindings and eval	✓	
Garbage collection	✓	
Tracing and debugging	✓	Always enabled
ObjectSpace	✓	Always enabled
Method invalidation	✓	
Constant invalidation	✓	

Charles Nutter: ‘So You Want to Optimize Ruby’ http://blog.headius.com/2012/10/so-you-want-to-optimize-ruby.html

Setup

- `git clone https://github.com/jruby/jruby.git`
- `cd jruby`
- mvn package

- Or just Google ‘jruby truffle wiki’

Fixnum to Bignum Promotion

- `Fixnum` – fixed integer: C `int64_t` or Java `long`
- `Bignum` – arbitrary integer: C `mpz_t` or Java `BigInteger`

- `Fixnum` overflows to `Bignum`
- `Bignum` underflows (?) to `Fixnum`
- Entirely different classes – programmer can tell the difference

- Unlike JavaScript and Python

Closures

- Anonymous functions that capture a lexical scope
- Called ‘blocks’ in Ruby – higher order methods

```
x = 14

my_array = [1, 2, 3, 4]

my_array.each do |n|
  puts x + n
end
```

Closures

- Anonymous functions that capture a lexical scope
- Called ‘blocks’ in Ruby – higher order methods

```
x = 14;

my_array = [1, 2, 3, 4];

my_array.each(function(n) {
  console.log(x + n);
});
```

Bindings and Eval

- Binding: get an environment as an object
- eval: as you'd expect, also lets you supply a Binding

```
def foo
  a = 1
  b = 2
  binding
end

puts foo.local_variable_get(:a)
```

Bindings and Eval

- Binding: get an environment as an object
- eval: as you'd expect, also lets you supply a Binding

```
alias :secret_binding :binding

def foo
  a = 1
  b = 2
  secret_binding
end

puts foo.local_variable_get(:a)
```

Method and Constant Invalidation

- Ruby lets you define methods – 'monkey patching'

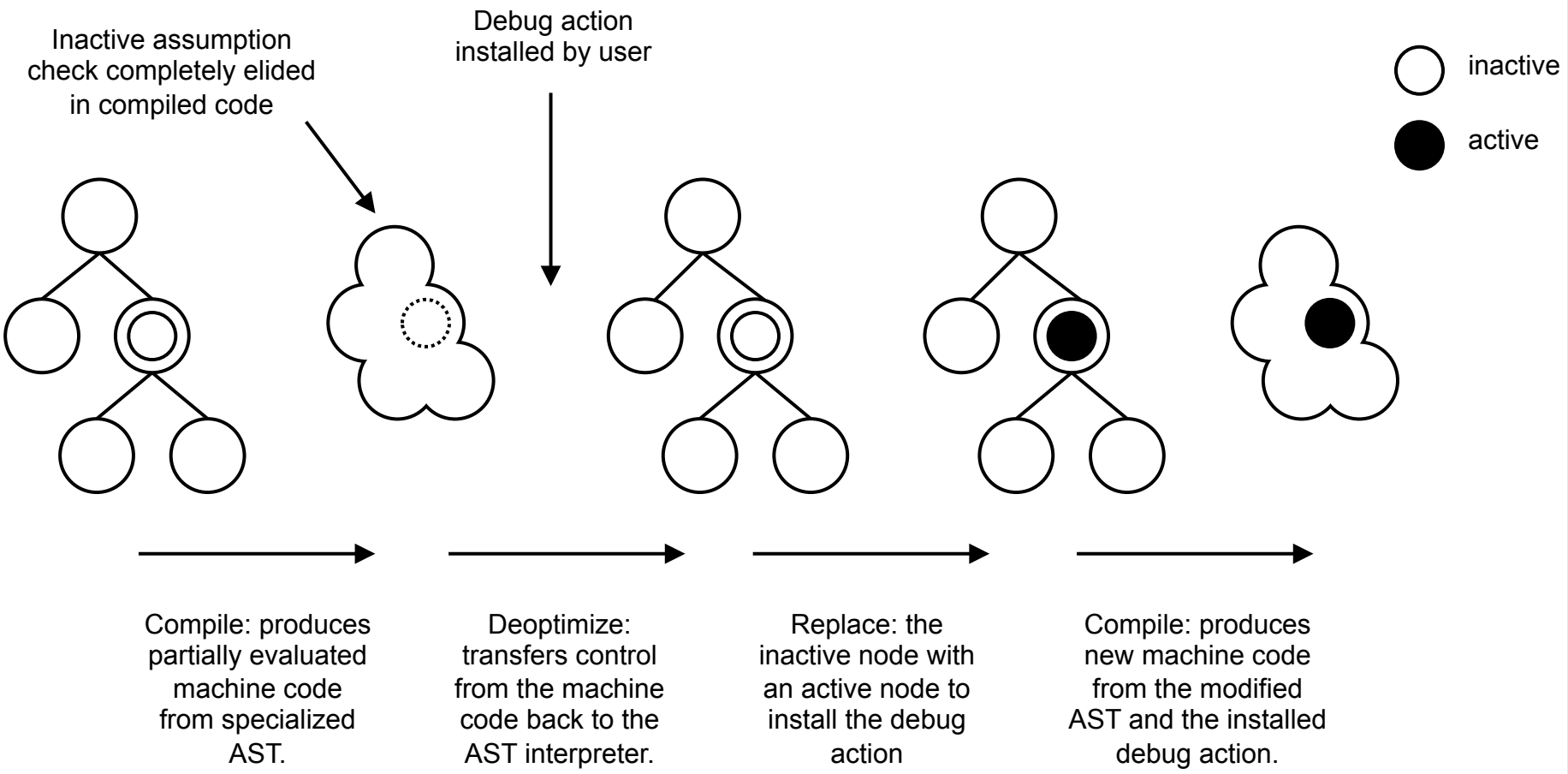
```
class Fixnum
  def *(b)
    self + b
  end
end

puts 14 * 2 => 16 (not 28)
```

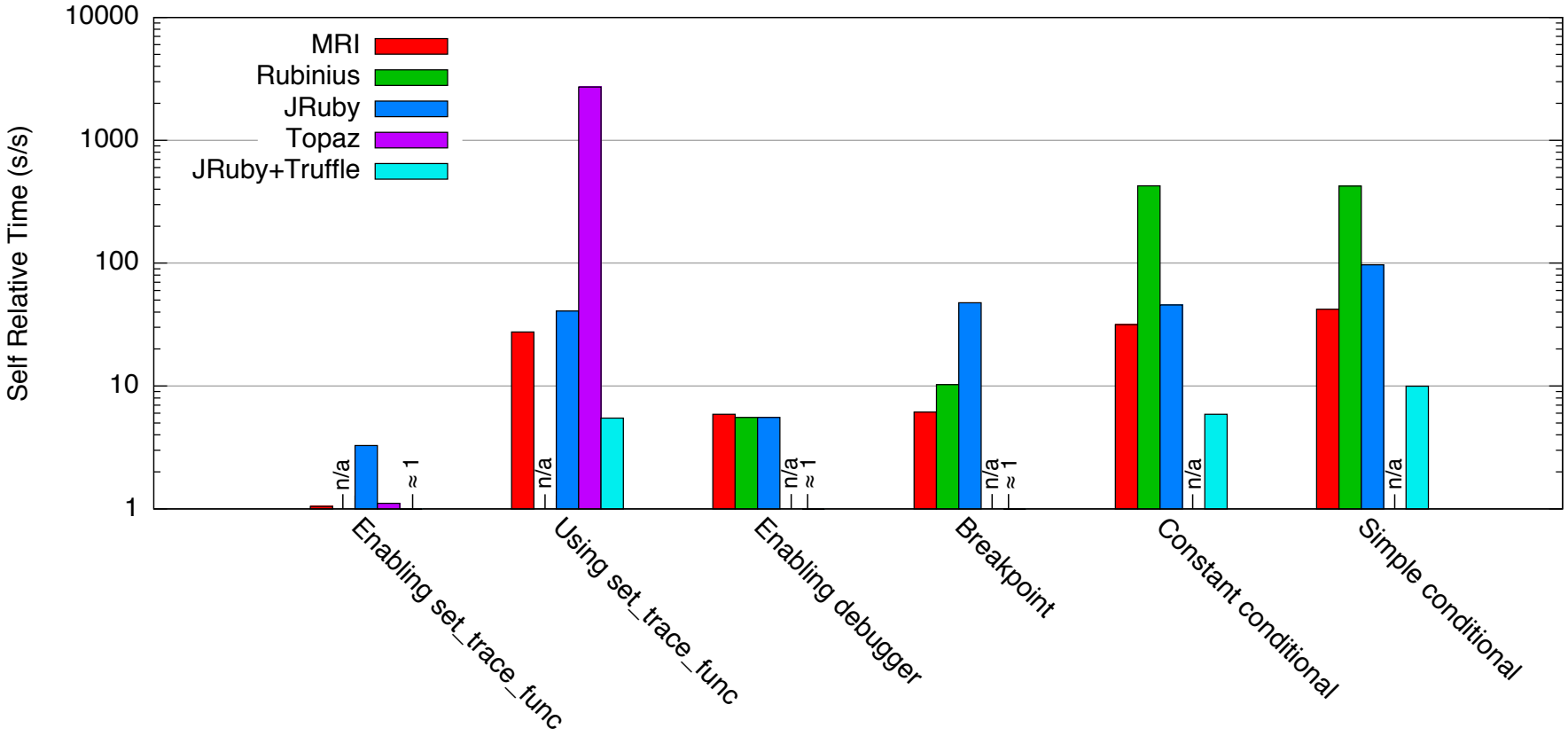
```
class Fixnum
  def *(b)
    eval "
      class Object::Fixnum
        def /(b)
          self - b
        end
      end
    "
    self + b
  end
end

puts 14 * 2 / 4 => 12 (not 4 or 7)
```

Tracing and Debugging

- `set_trace_func` allows you to install a method to be run on every line
- Behind a –debug flag in JRuby, not supported in Rubinius

```
set_trace_func proc { |line, binding|
  puts “We’re at line number #{line}”
}

x = 1  => “We’re at line number 6”
y = 2  => “We’re at line number 7”
```





ObjectSpace

- ObjectSpace allows you to enumerate all live objects
- Behind a flag –X+O in JRuby
- How to find all live objects in a JVM?

```
ObjectSpace#each_object do |o|
  puts o
end
```

Wrap Up

Get Involved

- Now is a great time to get involved in Truffle and Graal
- Personal opinion: I'd like to see them in JDK 9 in about 2 years

- Areas open for research: concurrency, parallelism, heterogeneous offload, language interoperability
- Build your language research on top of Truffle and Graal
- Implement a language: Haskell, Erlang, Swift, Clojure, PHP
- Design and implement an entirely new language

Get Involved

- http://openjdk.java.net/projects/graal/
- graal-dev@openjdk.java.net

- Documentation admittedly a little bit limited so far
- Look at SL and Ruby
- chris.seaton@oracle.com
- @ChrisGSeaton

Many people behind Truffle and Graal

Oracle Labs
Danilo Ansaloni
Daniele Bonetta
Laurent Daynès
Erik Eckstein
Michael Haupt
Mick Jordan
Peter Kessler
Christos Kotselidis
David Leibs
Tom Rodriguez
Roland Schatz
Doug Simon
Lukas Stadler
Michael Van De Vanter
Christian Wimmer
Christian Wirth
Mario Wolczko
Thomas Würthinger
Laura Hill (Manager)

Interns
Shams Imam
Stephen Kell
Gregor Richards
Rifat Shariyar

JKU Linz
Prof. Hanspeter Mössenböck
Gilles Duboscq
Matthias Grimmer
Christian Häubl
Josef Haider
Christian Humer
Christian Huber
Manuel Rigger
Bernhard Urban
Andreas Wöß

University of Manchester
Chris Seaton

University of Edinburgh
Christophe Dubach
Juan José
Fumero Alfonso
Ranjeet Singh
Toomas Remmelg

LaBRI
Floréal Morandat

University of California, Irvine
Prof. Michael Franz
Codrut Stancu
Gulfem Savrun Yeniceri
Wei Zhang

Purdue University
Prof. Jan Vitek
Tomas Kalibera
Petr Maj
Lei Zhao

T. U. Dortmund
Prof. Peter Marwedel
Helena Kotthaus
Ingo Korb

University of California, Davis
Prof. Duncan Temple Lang
Nicholas Ulle

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


ORACLE®