ORACLE®

JavaOne™

ORACLE®

Guilt Free Ruby on the JVM

Forgetting Conventional Wisdom

Chris Seaton
Research Manager
Oracle Labs
October 28, 2015

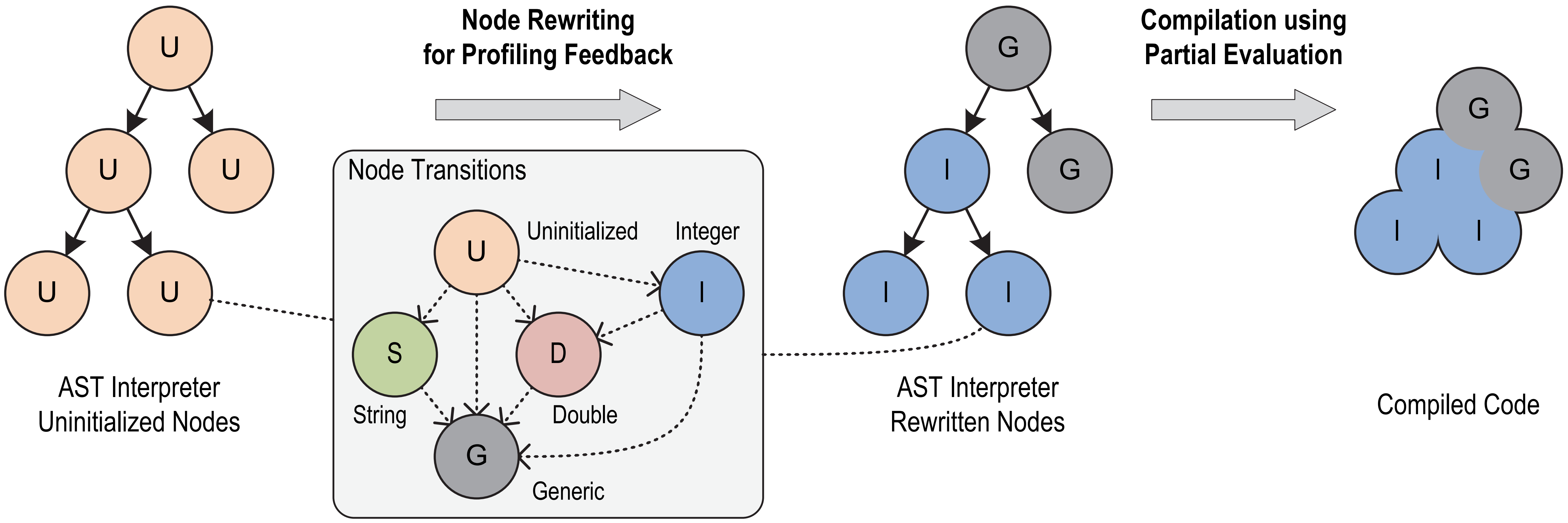
Safe Harbor Statement

Program Agenda

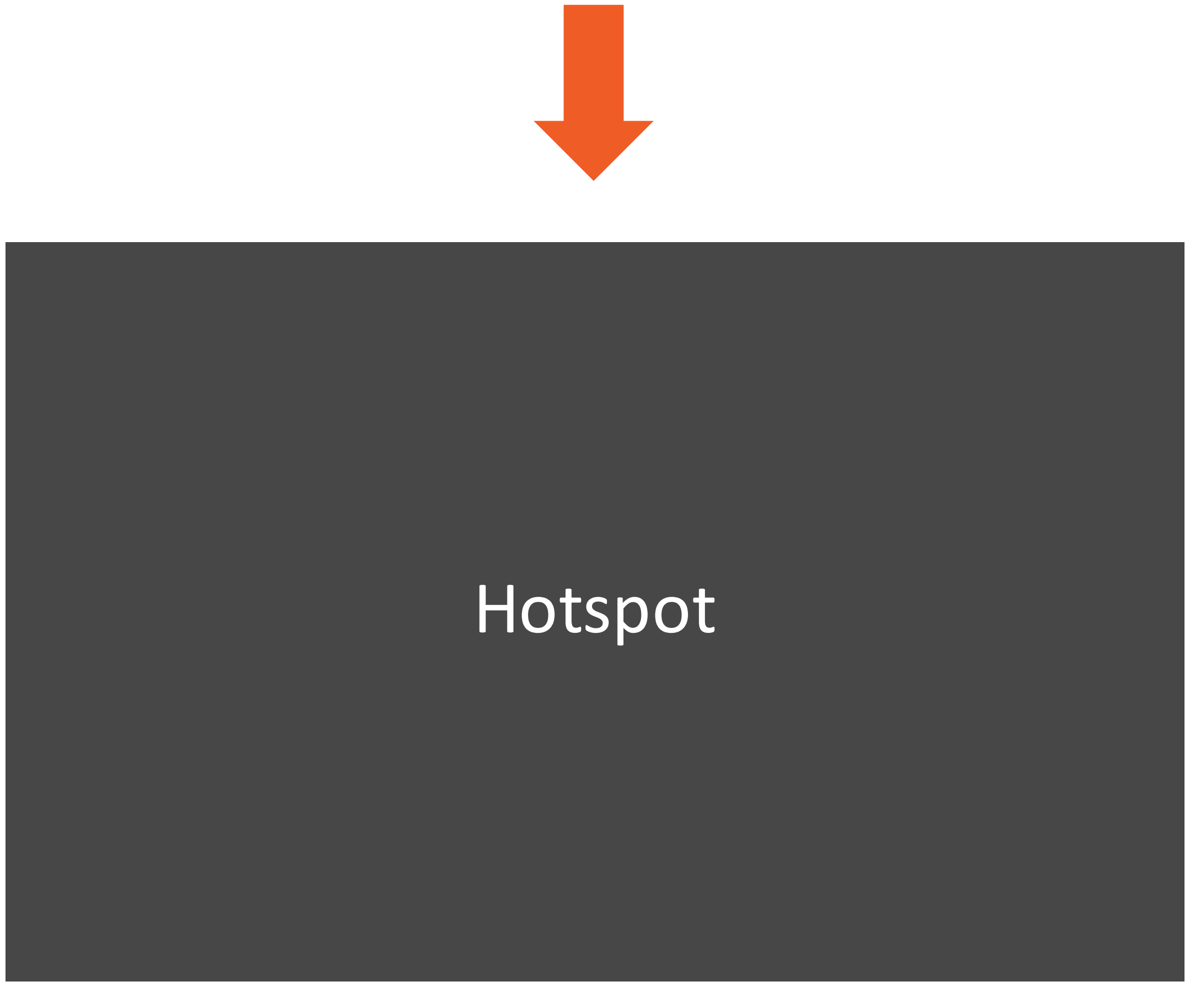
1. Ruby on the Graal VM
2. How do people want to write Ruby?
3. Why would they feel guilty abut it?
4. How can we fix that?
5. Optimisation deep dive

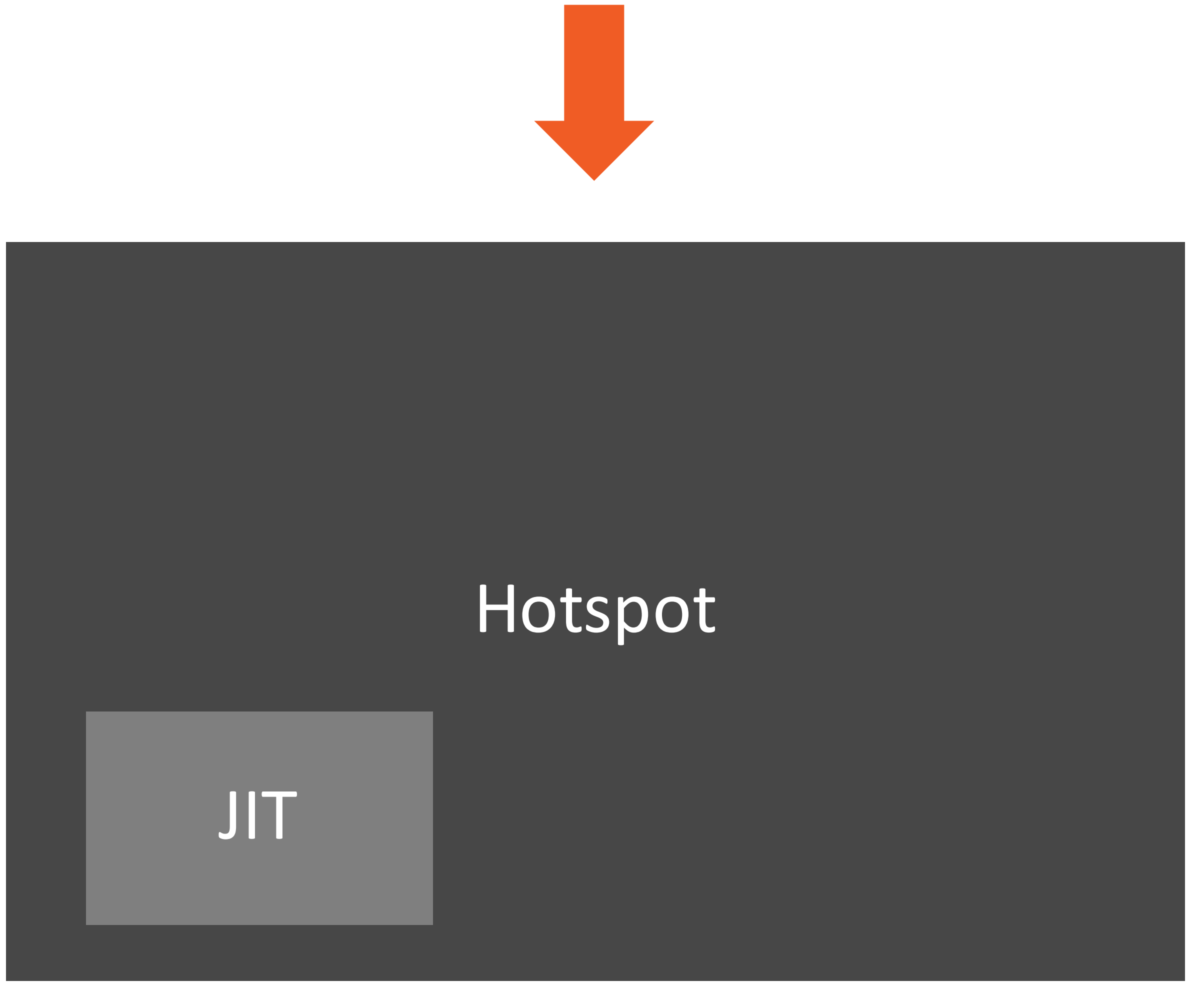
Ruby on the Graal VM

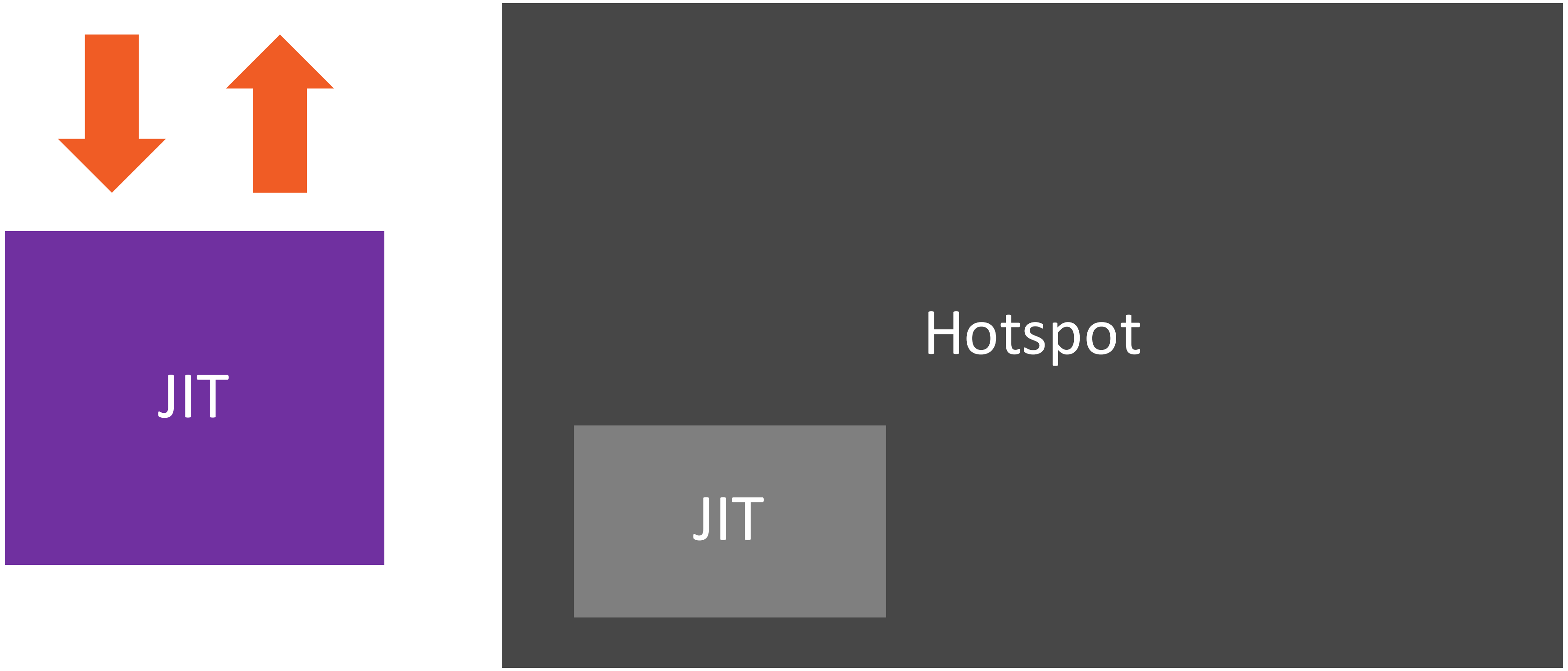
T. Würthinger, C. Wimmer, A. Wöß, L. Stadler, G. Duboscq, C. Humer, G. Richards, D. Simon, and M. Wolczko, “One VM to rule them all,” presented at the Onward! '13: Proceedings of the 2013 ACM international symposium on New ideas, new paradigms, and reflections on programming & software, New York, New York, USA, 2013, pp. 187–204.



Node Rewriting for Profiling Feedback
Compilation using Partial Evaluation
U
U
U
U
U
AST Interpreter Uninitialized Nodes
Node Transitions
U
Uninitialized
Integer
I
S
String
D
Double
G
Generic
G
I
G
I
I
AST Interpreter Rewritten Nodes
G
I
G
I
I
Compiled Code

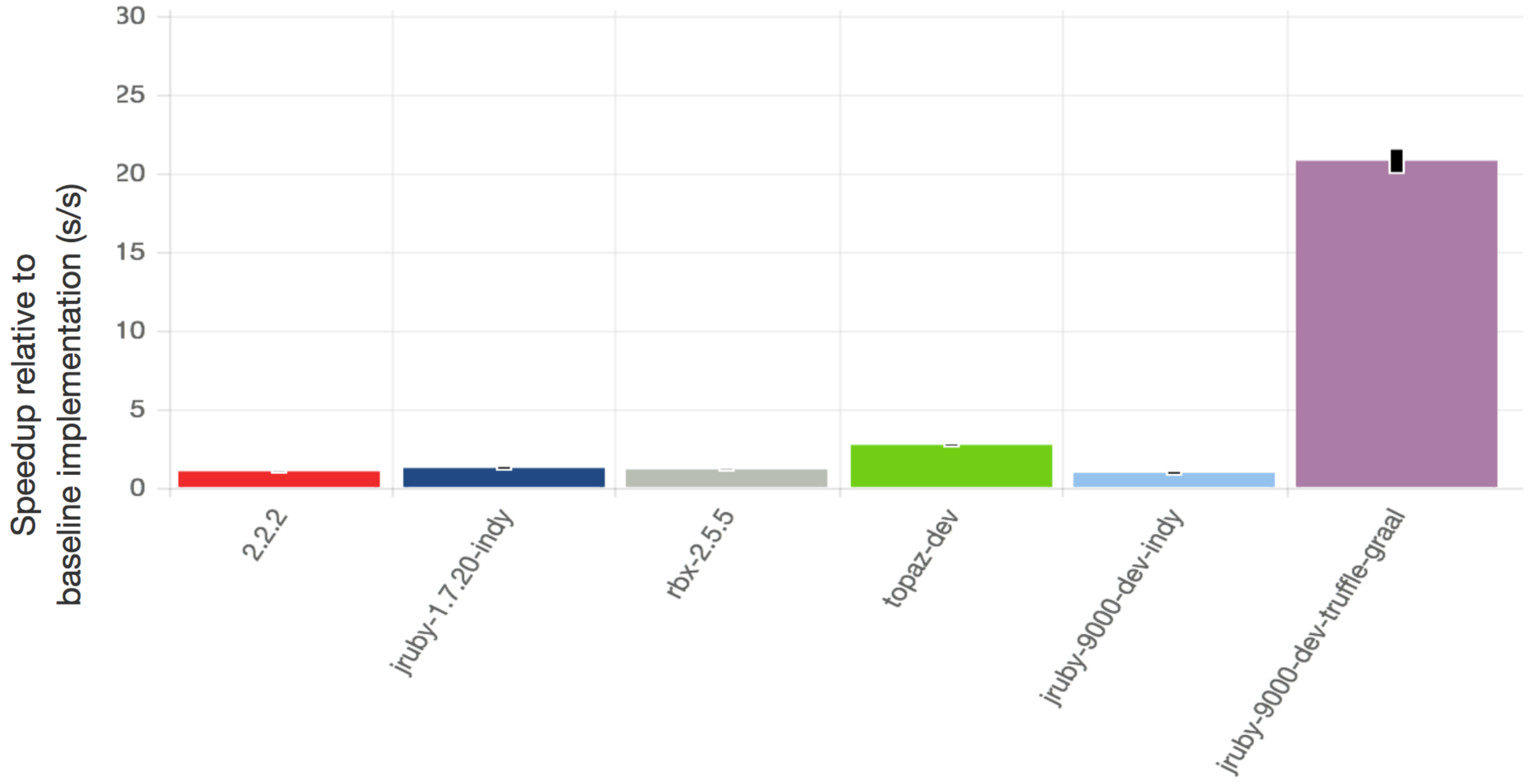

Hotspot


Hotspot
JIT


JIT
Hotspot
JIT

90%

Compatibility with the language and core library


Speedup relative to baseline implementation (s/s)
30
25
20
15
10
5
0
2.2.2
jruby-1.7.20-indy
rbx-2.5.5
topaz-dev
jruby-9000-dev-indy
jruby-9000-dev-truffle-graal

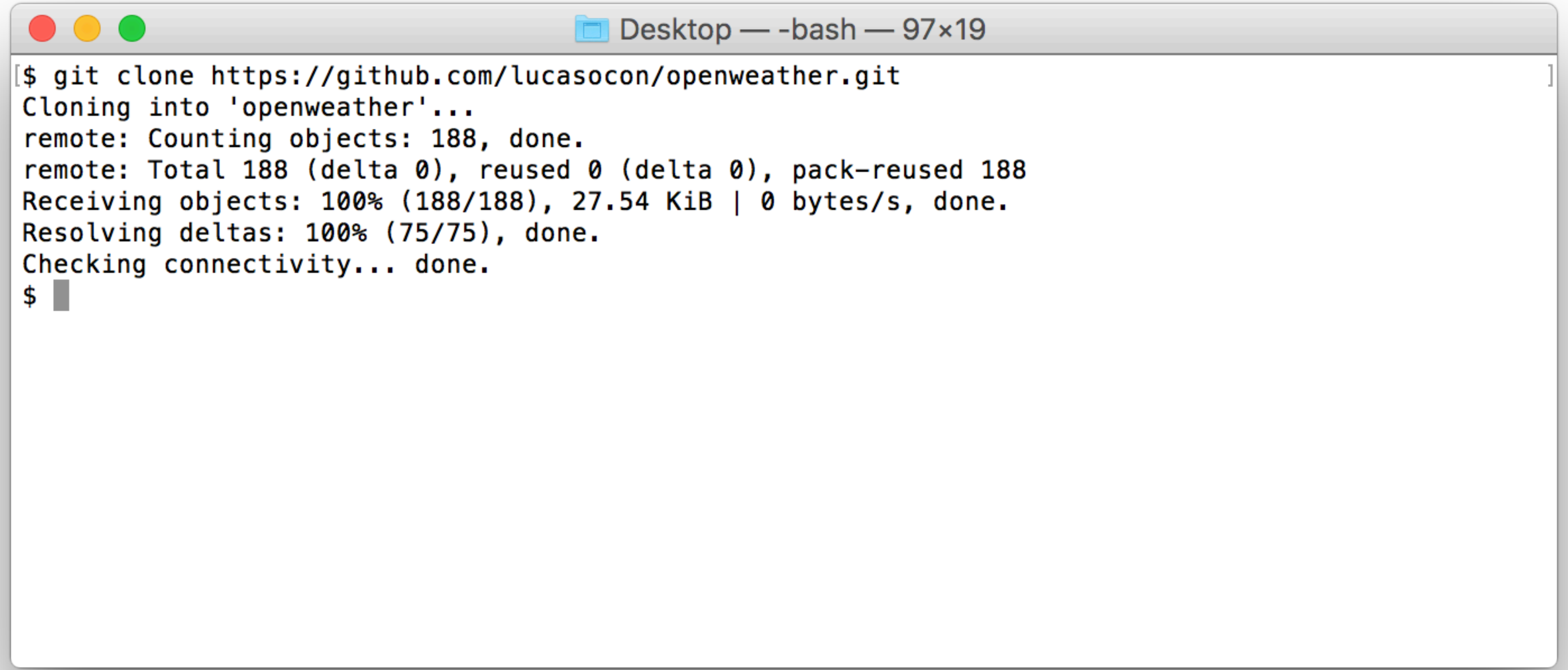
Can I run real code today?

http://goo.gl/ssTA2T

(or search for ‘graal’ on the Oracle Technology Network)

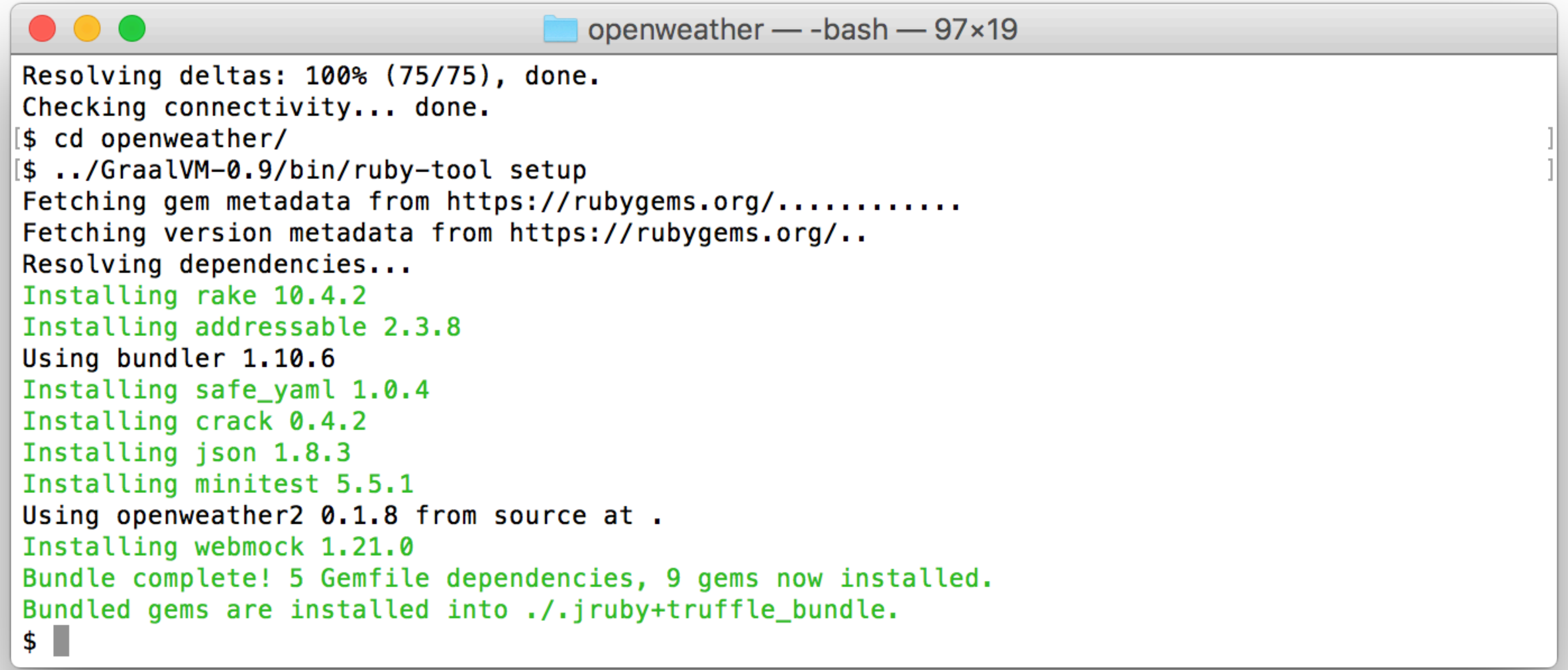
rbenv install jruby-master+graal-dev

git clone https://github.com/lucasocon/openweather.git



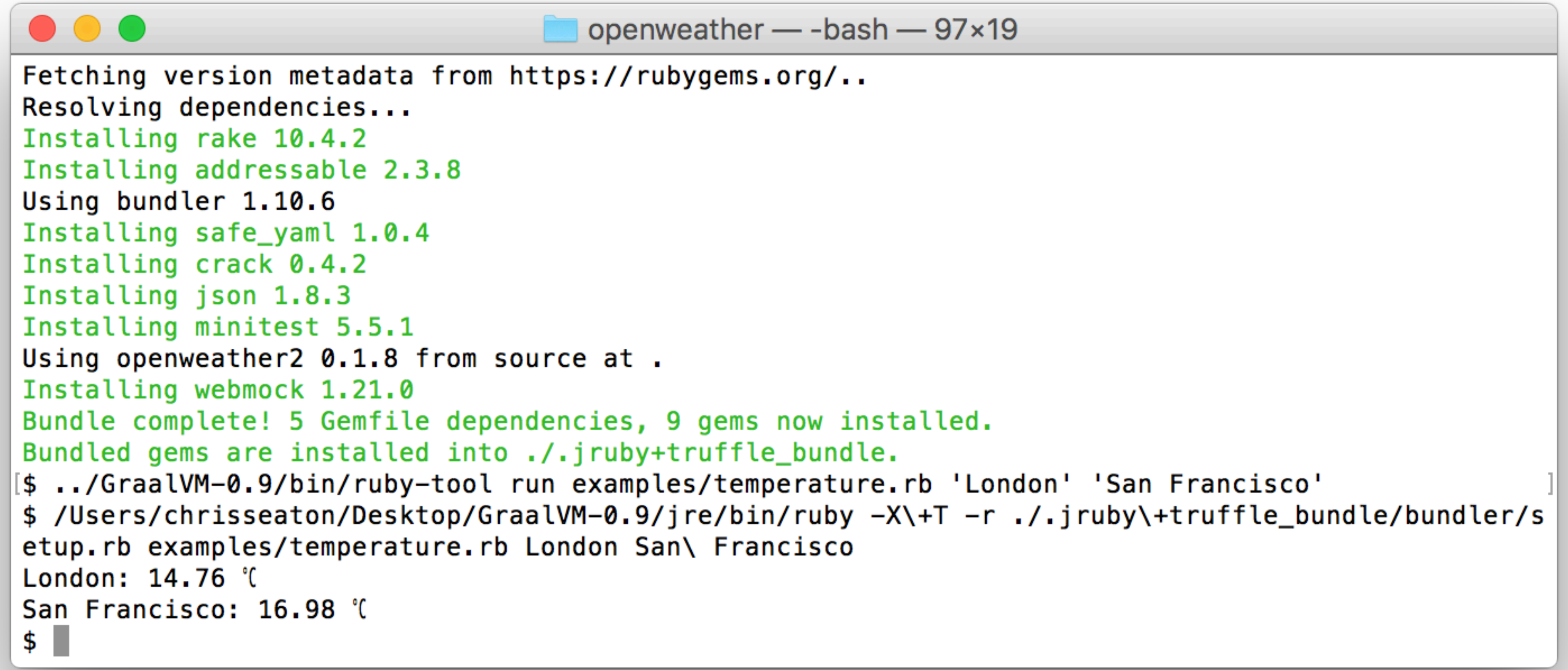
```
$ git clone https://github.com/lucasocon/openweather.git
Cloning into 'openweather'...
remote: Counting objects: 188, done.
remote: Total 188 (delta 0), reused 0 (delta 0), pack-reused 188
Receiving objects: 100% (188/188), 27.54 KiB | 0 bytes/s, done.
Resolving deltas: 100% (75/75), done.
Checking connectivity... done.
$ 
```

../GraalVM-0.9/bin/ruby-tool setup



```
Resolving deltas: 100% (75/75), done.
Checking connectivity... done.
[$ cd openweather/
[$ ../GraalVM-0.9/bin/ruby-tool setup
Fetching gem metadata from https://rubygems.org/............
Fetching version metadata from https://rubygems.org/..
Resolving dependencies...
Installing rake 10.4.2
Installing addressable 2.3.8
Using bundler 1.10.6
Installing safe_yaml 1.0.4
Installing crack 0.4.2
Installing json 1.8.3
Installing minitest 5.5.1
Using openweather2 0.1.8 from source at .
Installing webmock 1.21.0
Bundle complete! 5 Gemfile dependencies, 9 gems now installed.
Bundled gems are installed into ./.jruby+truffle_bundle.
$
```

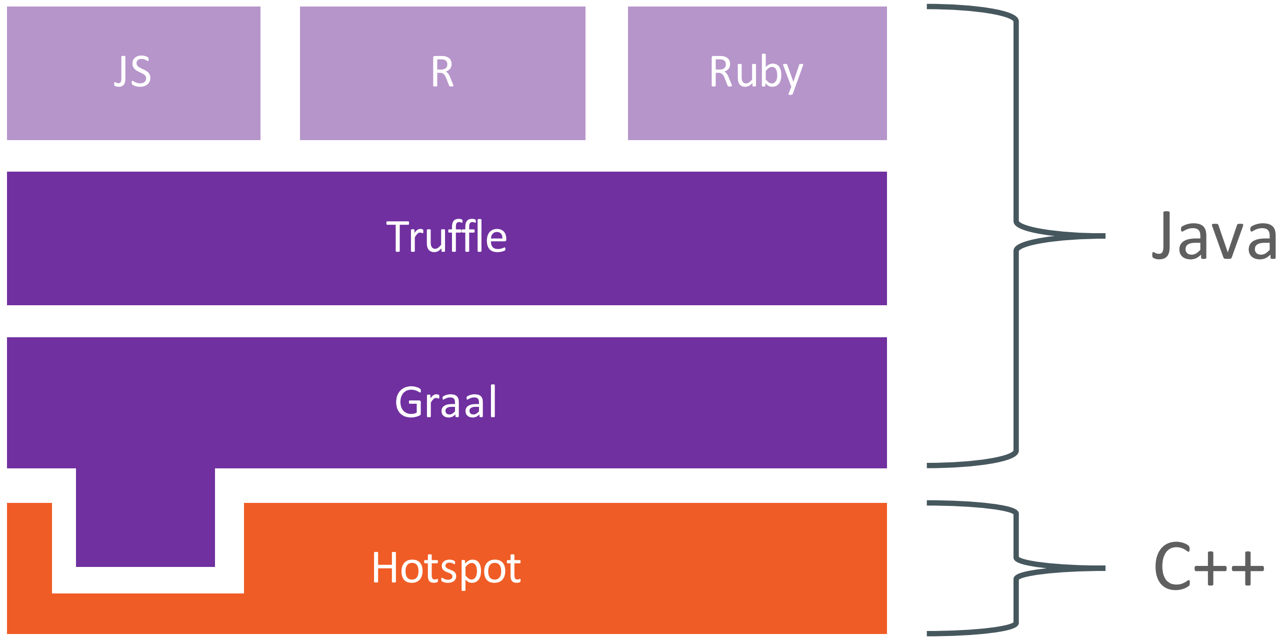
../GraalVM-0.9/bin/ruby-tool run London ‘San Francisco’

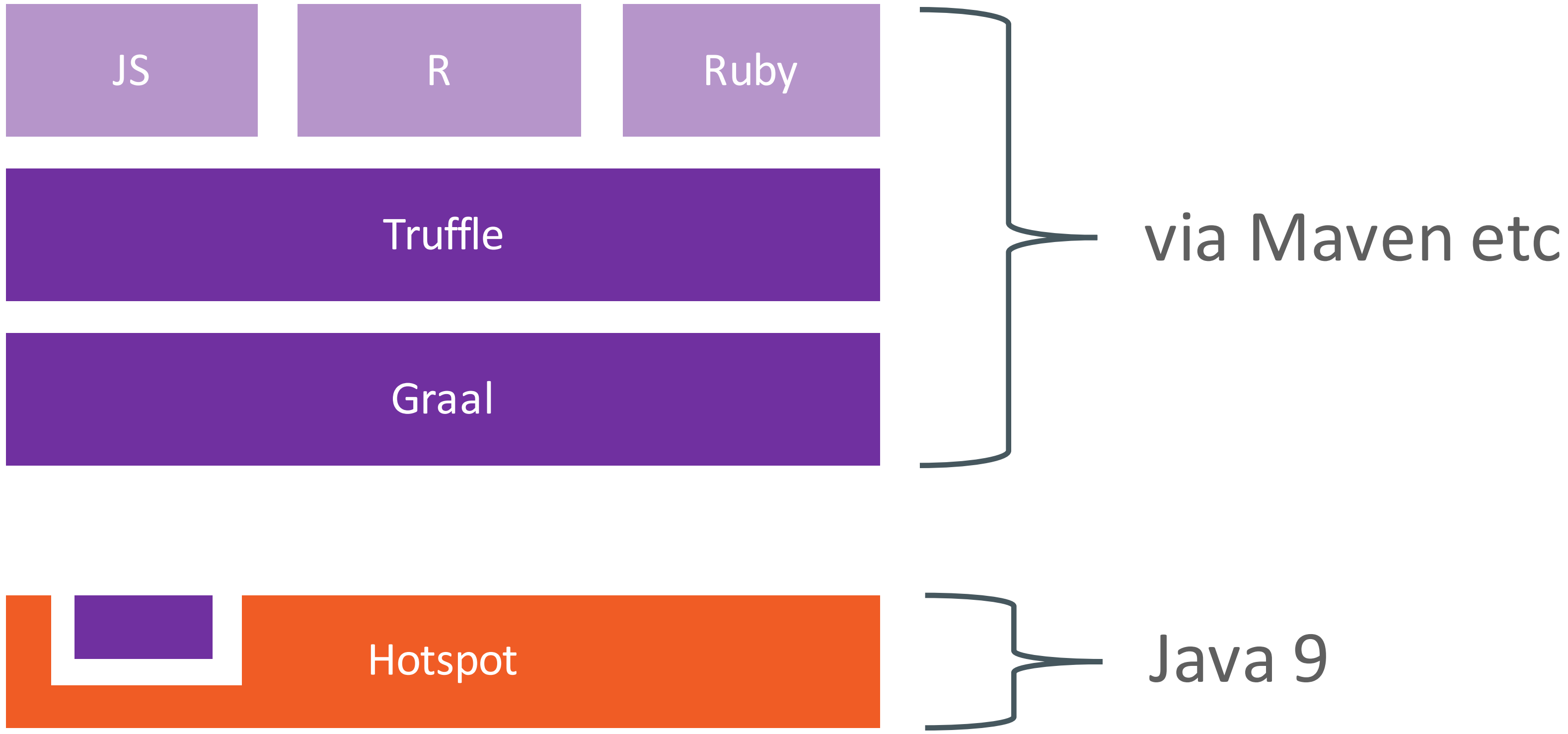

openweather — -bash — 97×19

```
Fetching version metadata from https://rubygems.org/..
Resolving dependencies...
Installing rake 10.4.2
Installing addressable 2.3.8
Using bundler 1.10.6
Installing safe_yaml 1.0.4
Installing crack 0.4.2
Installing json 1.8.3
Installing minitest 5.5.1
Using openweather2 0.1.8 from source at .
Installing webmock 1.21.0
Bundle complete! 5 Gemfile dependencies, 9 gems now installed.
Bundled gems are installed into ./.jruby+truffle_bundle.
$ ../GraalVM-0.9/bin/ruby-tool run examples/temperature.rb 'London' 'San Francisco'
$ /Users/chrisseaton/Desktop/GraalVM-0.9/jre/bin/ruby -X\+T -r ./.jruby\+truffle_bundle/bundler/s
etup.rb examples/temperature.rb London San\ Francisco
London: 14.76 ℃
San Francisco: 16.98 ℃
$
```

Will it be supported in standard Java?

JVMCI
(JVM Compiler Interface)




JS
R
Ruby
Truffle
Graal
Hotspot
via Maven etc
Java 9

How do people want to write Ruby?

```
class Object
  # An object is blank if it's false, empty, or a whitespace string.
  # For example, '', '   ', +nil+, [], and {} are all blank.
  def blank?
    respond_to?(:empty?) ? !!empty? : !self
  end
end
```

```
def hard_mix(fg, bg, opts={})
  return apply_opacity(fg, opts)
    if fully_transparent?(bg)

  return bg if fully_transparent?(fg)

  mix_alpha, dst_alpha = calculate_alphas(
    fg, bg, DEFAULT_OPTS.merge(opts))

  new_r = blend_channel(r(bg), (r(bg)
    + r(fg) <= 255) ? 0 : 255, mix_alpha)
  new_g = blend_channel(g(bg), (g(bg)
    + g(fg) <= 255) ? 0 : 255, mix_alpha)
  new_b = blend_channel(b(bg), (b(bg)
    + b(fg) <= 255) ? 0 : 255, mix_alpha)

  rgba(new_r, new_g, new_b, dst_alpha)
end

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

```
def grayscale_entry(bit_depth)
  value = ChunkyPNG::Canvas.send(
    :"decode_png_resample_#{bit_depth}bit_value",
    content.unpack('n')[0])
  ChunkyPNG::Color.grayscale(value)
end
```

```ruby
class Duration
  attr_accessor :value

  def initialize(value)
    @value = value
  end

  def as_json
    ...
  end

  def inspect
    ...
  end

  def method_missing(method, *args, &block)
    value.send(method, *args, &block)
  end
end
```

```
def delegate(method)
  method_def = (
    "def #{method}(*args, &block)\n" +
    "  delegated.#{method}(*args, &block)\n" +
    "end"
  )
  module_eval(method_def, file, line)
end
```

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

Why would you feel guilty about this?

Need to C extensions to remove abstraction

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

Need to C extensions to remove abstraction

```
VALUE psd_native_util_clamp(VALUE self,
    VALUE r_num, VALUE r_min, VALUE r_max) {
  int num = FIX2INT(r_num);
  int min = FIX2INT(r_min);
  int max = FIX2INT(r_max);
  return num > max ?
      r_max
    : (num < min ? r_min : r_num);
}
```

Metaprogramming method calls are slow

```
14 + 2
14.send(:+, 2)
14.send('+', 2)
operator = '+'; 14.send("#{operator}", 2)
```

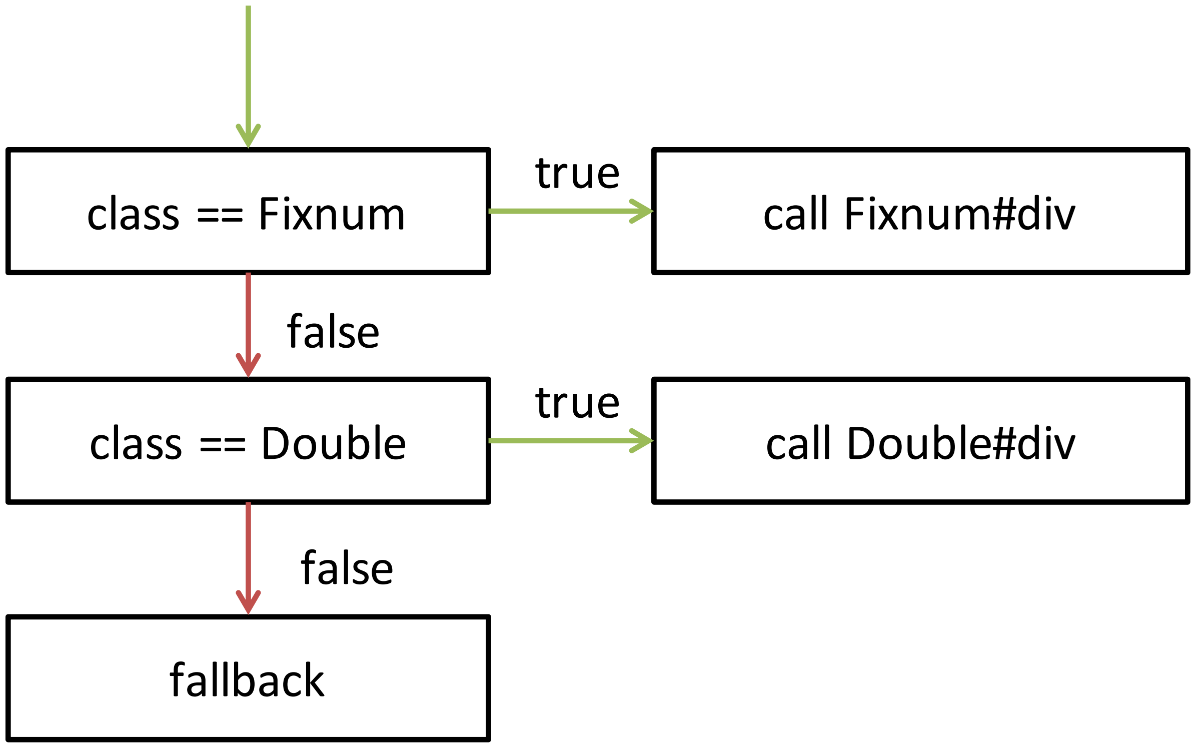
Eval is slow

```
14 + 2
eval('14 + 2')
x = 14; y = 2; eval("#{x} + #{y}")
```

Making Ruby as people use it fast

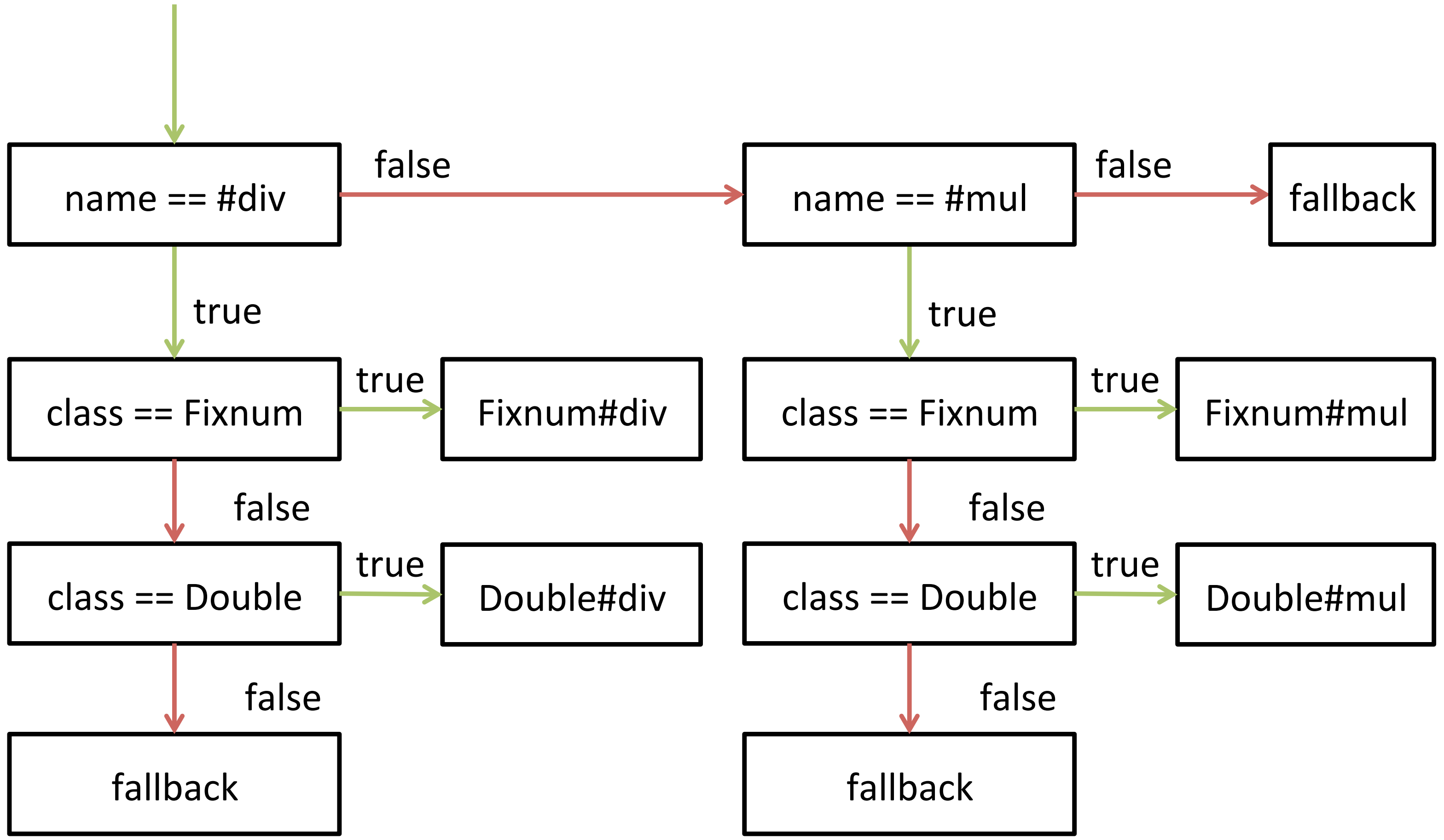
Method caching

```
14 / 2
14.0 / 2.0
```



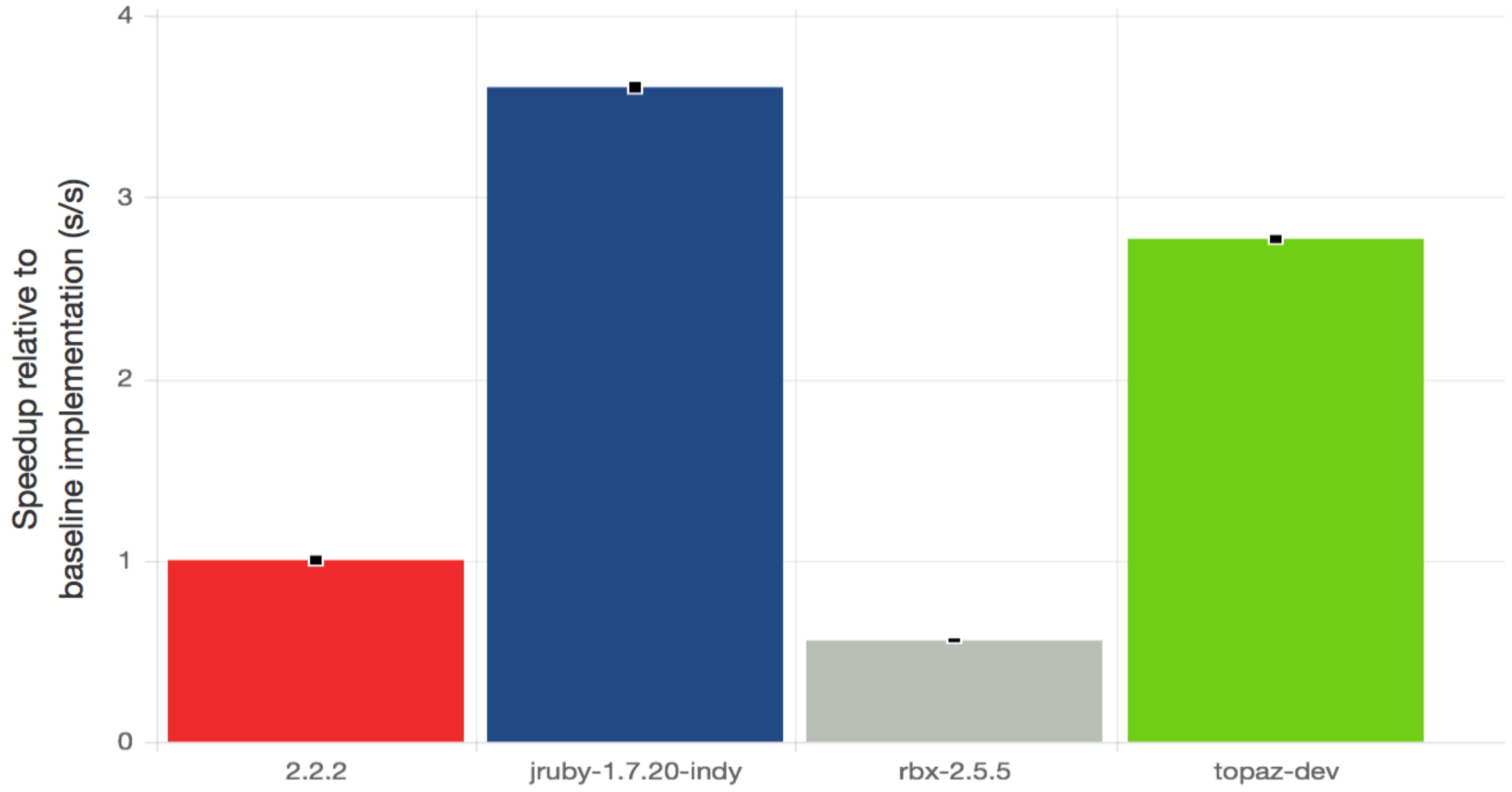
Metaprogramming method caching

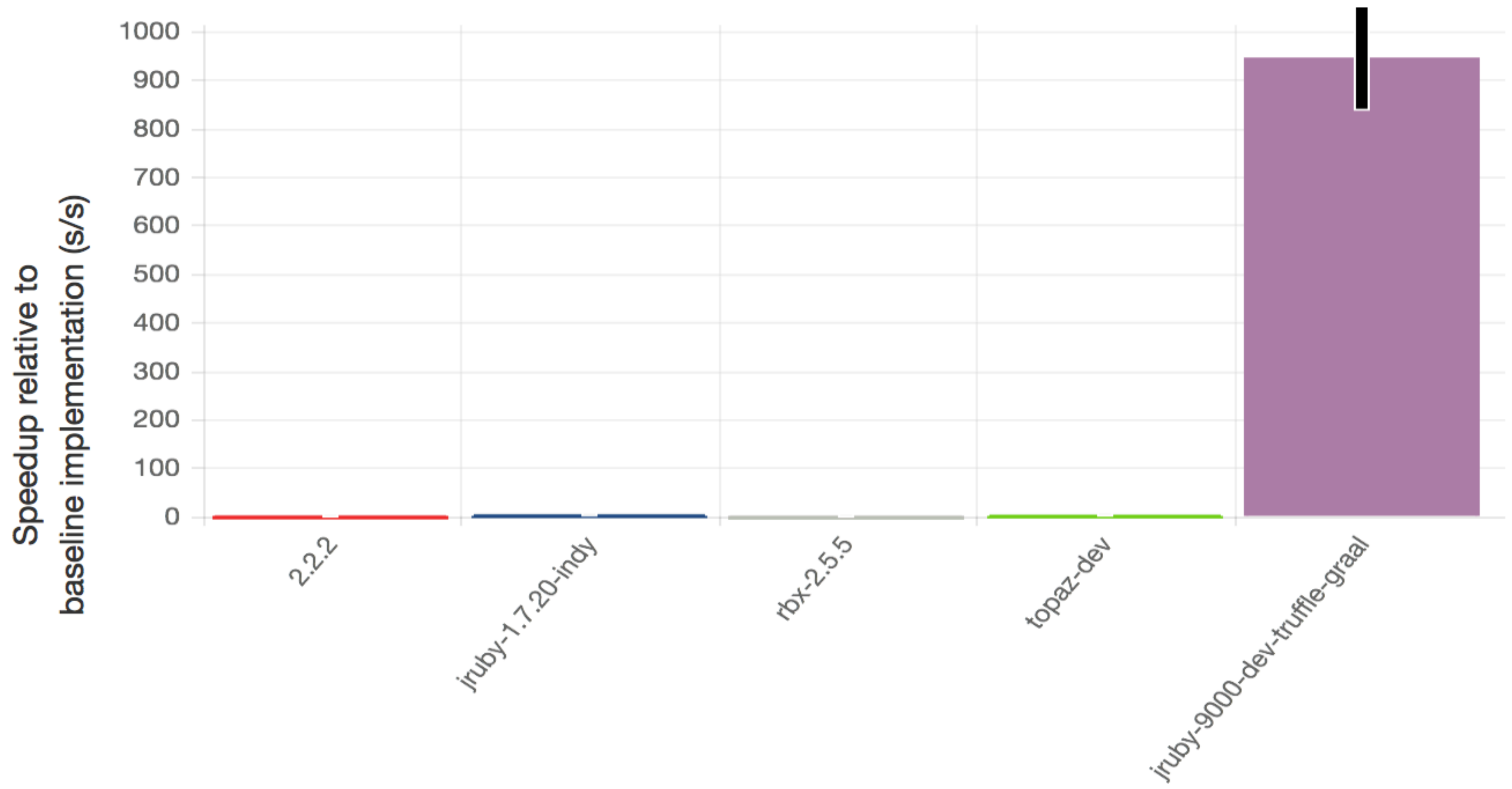
```
14.send(:/, 2)
14.0.send(:/, 2.0)
14.send(:*, 2)
14.0.send(:*, 2.0)
```

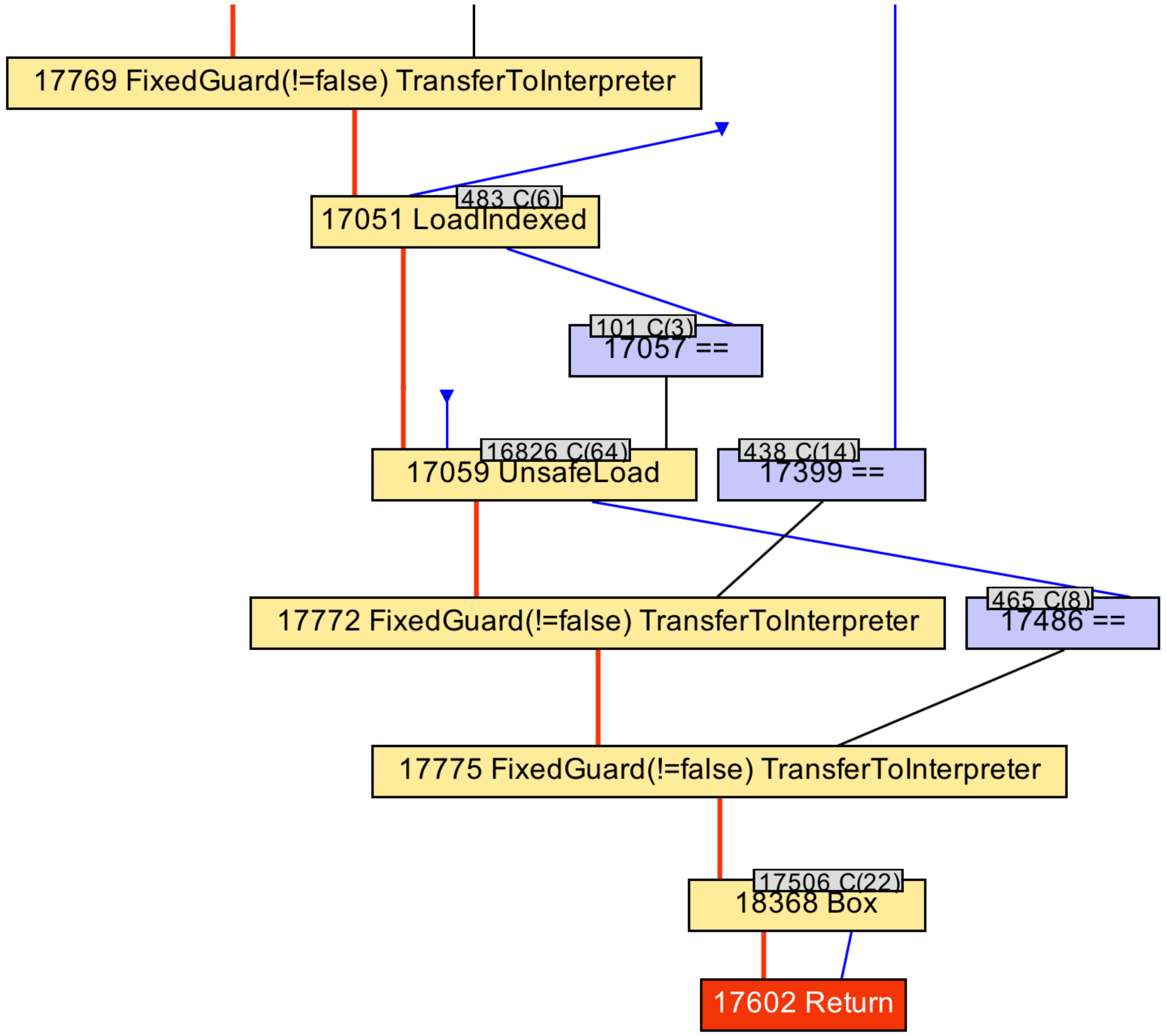


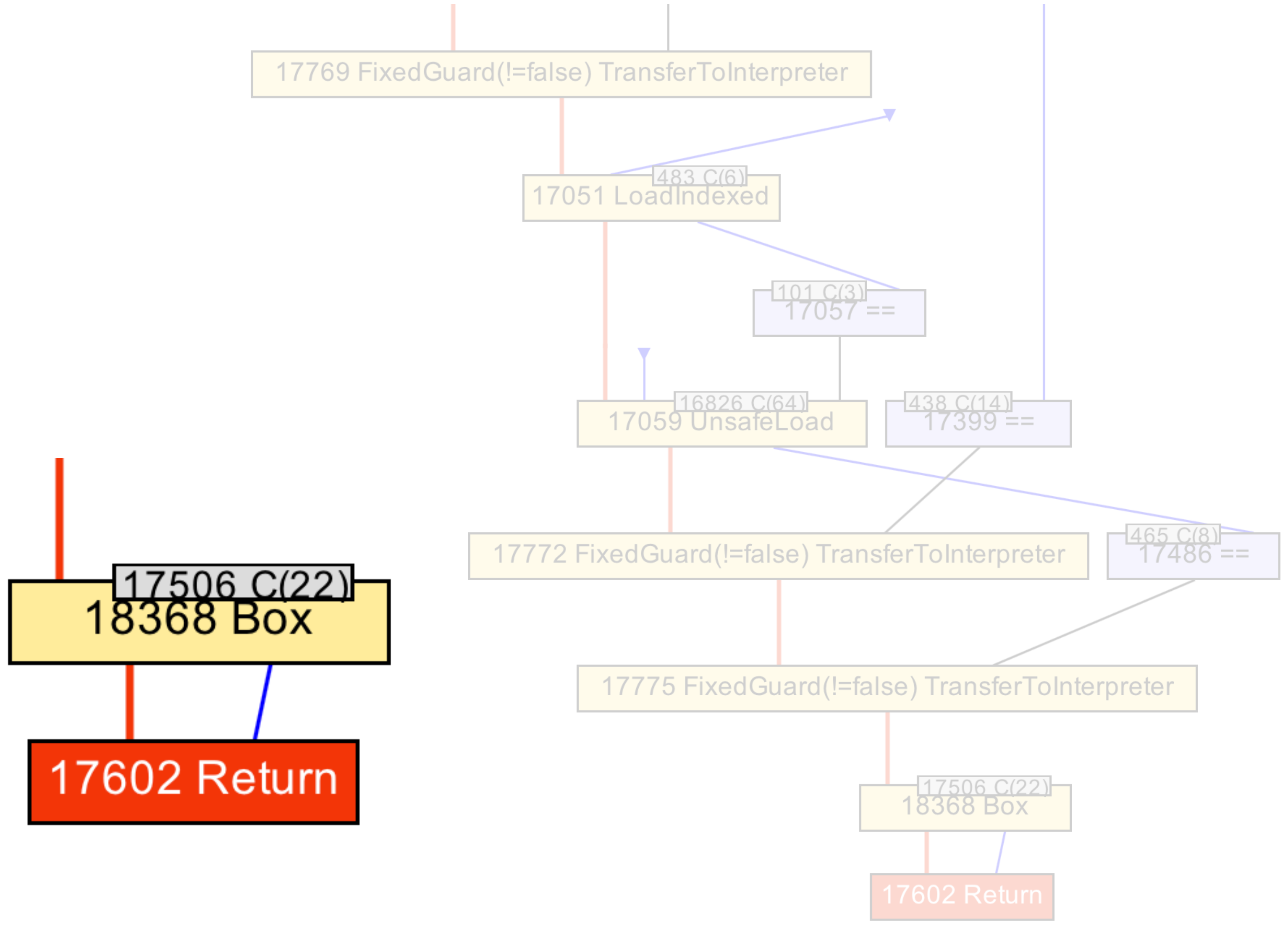
```ruby
module Foo
  def self.foo(a, b, c)
    hash = {a: a, b: b, c: c}
    array = hash.map { |k, v| v }
    x = array[0]
    y = [a, b, c].sort[1]
    x + y
  end
end

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

```ruby
bar = Bar.new

loop do
  start = Time.now
  1_000_000.times do
    bar.foo(14, 8, 6)
  end
  puts Time.now - start
end
```


Speedup relative to baseline implementation (s/s)
4
3
2
1
0
2.2.2
jruby-1.7.20-indy
rbx-2.5.5
topaz-dev


Speedup relative to baseline implementation (s/s)
1000
900
800
700
600
500
400
300
200
100
0
2.2.2
jruby-1.7.20-indy
rbx-2.5.5
topaz-dev
jruby-9000-dev-truffle-graal


17769 FixedGuard(!=false) TransferToInterpreter
483 C(6)
17051 LoadIndexed
101 C(3)
17057 ==
16826 C(64)
17059 UnsafeLoad
438 C(14)
17399 ==
17772 FixedGuard(!=false) TransferToInterpreter
465 C(8)
17486 ==
17775 FixedGuard(!=false) TransferToInterpreter
17506 C(22)
18368 Box
17602 Return


17769 FixedGuard(!=false) TransferToInterpreter
483 C(6)
17051 LoadIndexed
101 C(3)
17057 ==
16826 C(64)
17059 UnsafeLoad
438 C(14)
17399 ==
17772 FixedGuard(!=false) TransferToInterpreter
465 C(8)
17486 ==
17775 FixedGuard(!=false) TransferToInterpreter
17506 C(22)
18368 Box
17602 Return
17506 C(22)
18368 Box
17602 Return

```
...
movabs 0x11e2037a8, %rax ; {oop(a 'java/lang/Integer' = 22)}
...
retq
```

Can we fold yet?

Conclusions

We don’t have to pretend Ruby is Java in order to optimise it

Conclusions

We don’t have to pretend Ruby is Java in order to optimise it

We don’t need to tell Ruby programmers to avoid language features to get performance

chris.seaton@oracle.com

@ChrisGSeaton

https://github.com/jruby/jruby/wiki/Truffle

(or just search for ‘jruby truffle’)

Acknowledgements

Oracle Labs
Danilo Ansaloni
Stefan Anzinger
Daniele Bonetta
Matthias Brantner
Laurent Daynès
Gilles Duboscq
Michael Haupt
Christian Humer
Mick Jordan
Peter Kessler
Hyunjin Lee
David Leibs
Kevin Menard
Tom Rodriguez
Roland Schatz
Chris Seaton
Doug Simon
Lukas Stadler
Michael Van de Vanter

Oracle Labs (continued)
Adam Welc
Till Westmann
Christian Wimmer
Christian Wirth
Paul Wögerer
Mario Wolczko
Andreas Wöß
Thomas Würthinger

Oracle Labs Interns
Shams Imam
Stephen Kell
Gero Leinemann
Julian Lettner
Gregor Richards
Robert Seilbeck
Rifat Shariyar

Oracle Labs Alumni
Erik Eckstein
Christos Kotselidi

JKU Linz
Prof. Hanspeter Mössenböck
Benoit Daloze
Josef Eisl
Thomas Feichtinger
Matthias Grimmer
Christian Häub
Josef Haider
Christian Hube
David Leopoltsederr
Manuel Rigger
Stefan Rumzucker
Bernhard Urban

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
Romand Tsegelskyi
Prahlad Joshi
Petr Maj Lei Zhao

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

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

Questions?

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.

Integrated Cloud

Applications & Platform Services

JavaOne™

ORACLE®

ORACLE®