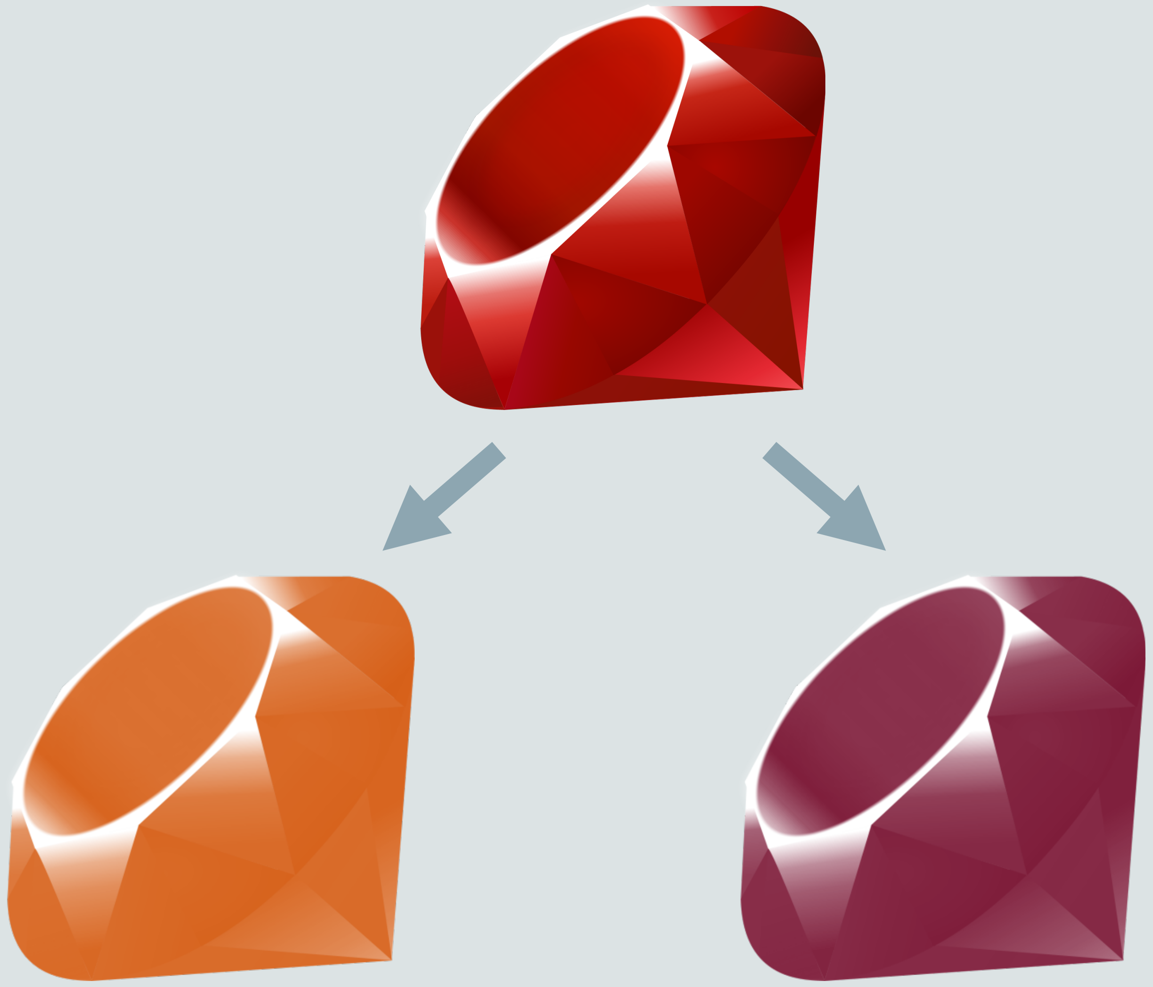
Faster Ruby and JS with Graal/Truffle



Chris Seaton
Oracle Labs

@ChrisGSeaton

11 April 2016

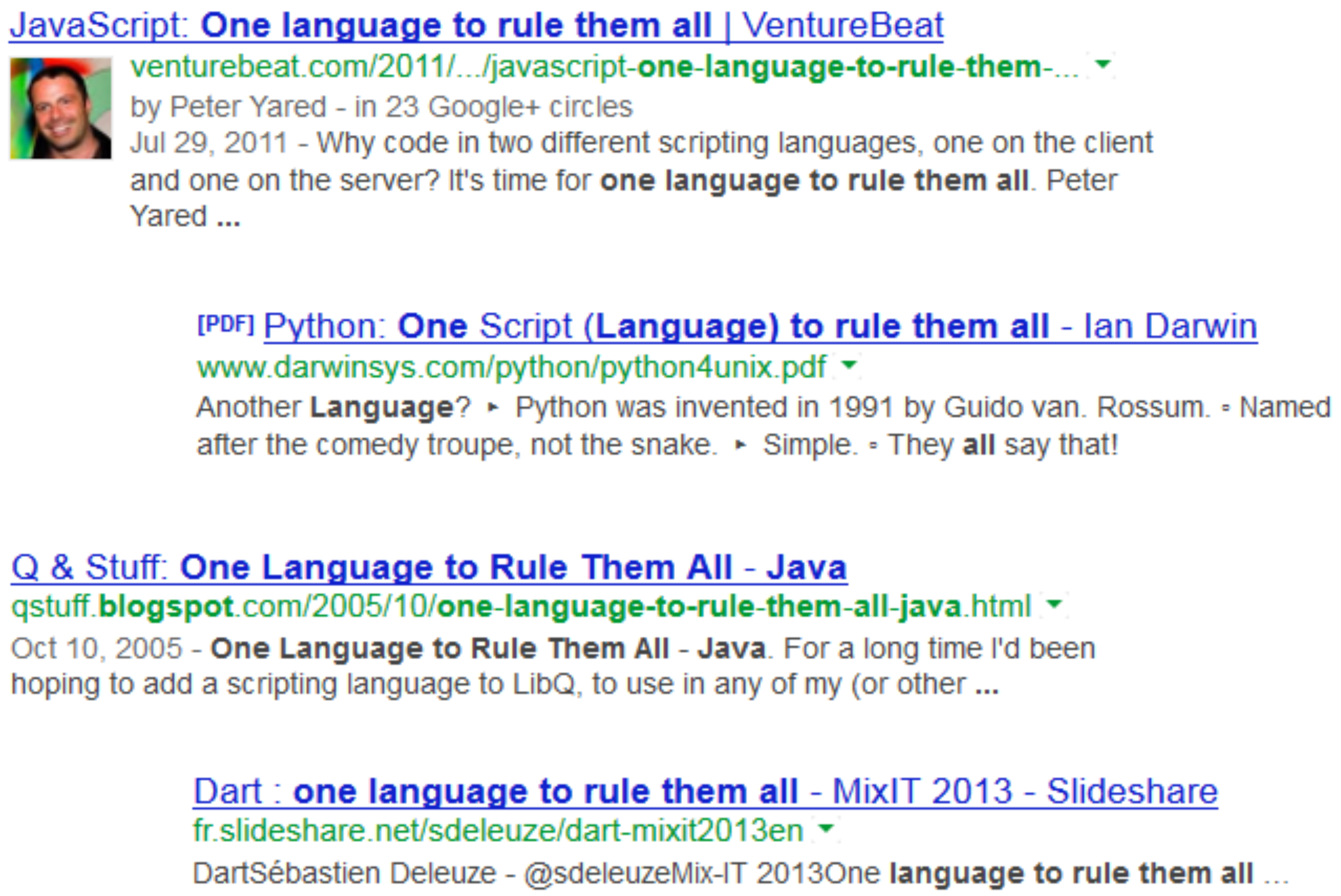
ORACLE

Safe Harbor Statement

ORACLE

ORACLE

One virtual machine to rule them all

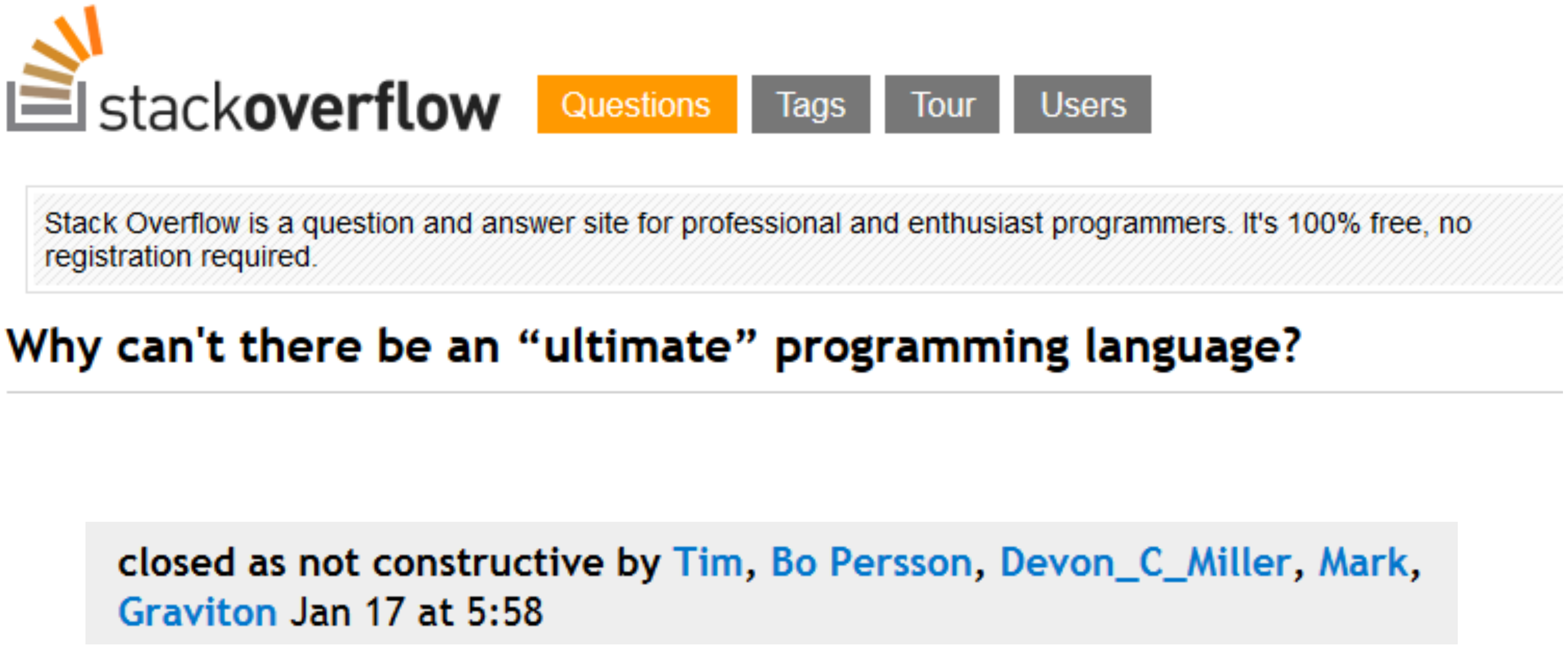


JavaScript: **One language to rule them all** | VentureBeat
venturebeat.com/2011/.../javascript-**one-language-to-rule-them**-...
by Peter Yared - in 23 Google+ circles
Jul 29, 2011 - Why code in two different scripting languages, one on the client and one on the server? It's time for **one language 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** ...

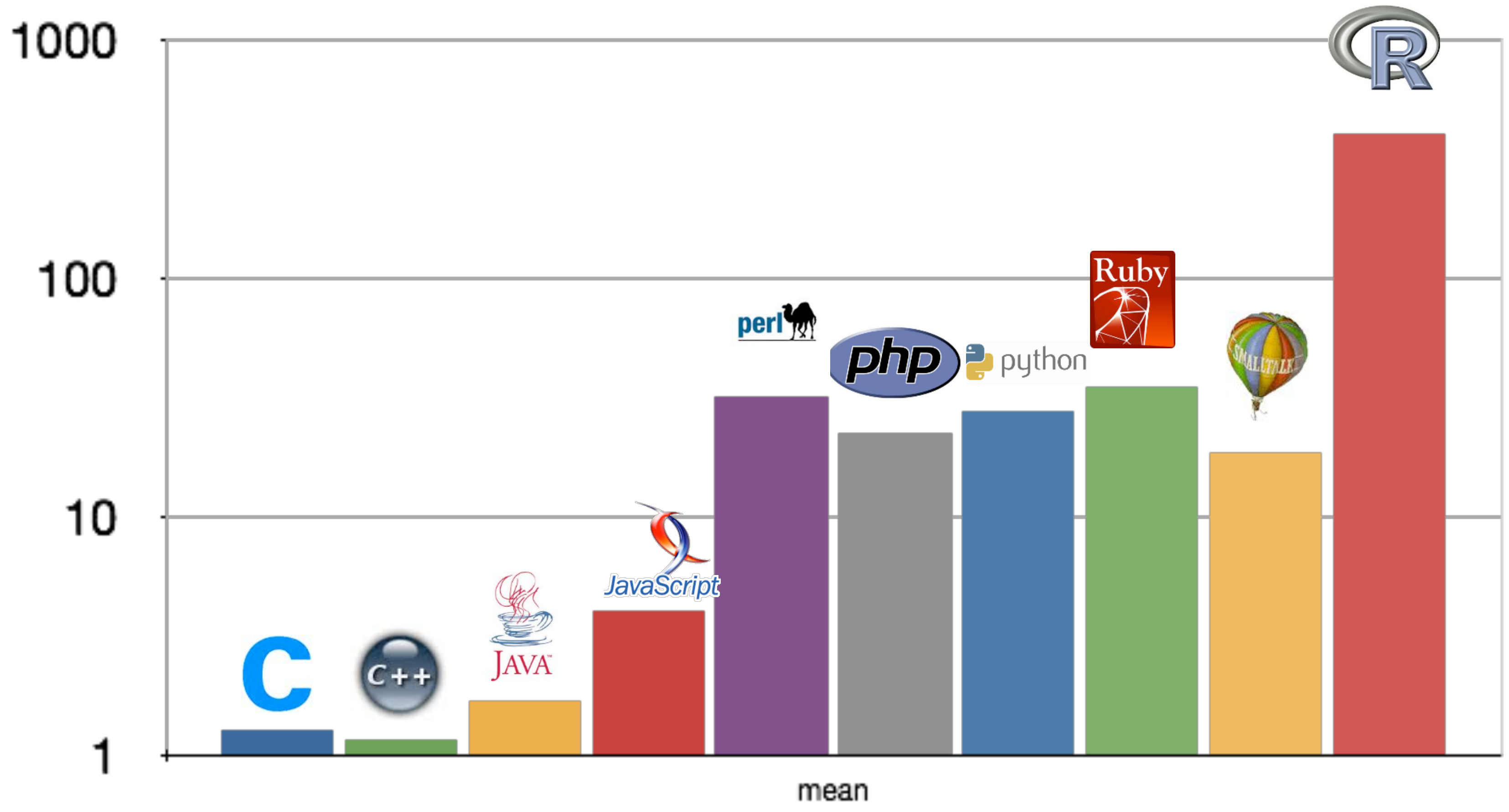


Stack Overflow is a question and answer site for professional and enthusiast programmers. It's 100% free, no registration required.

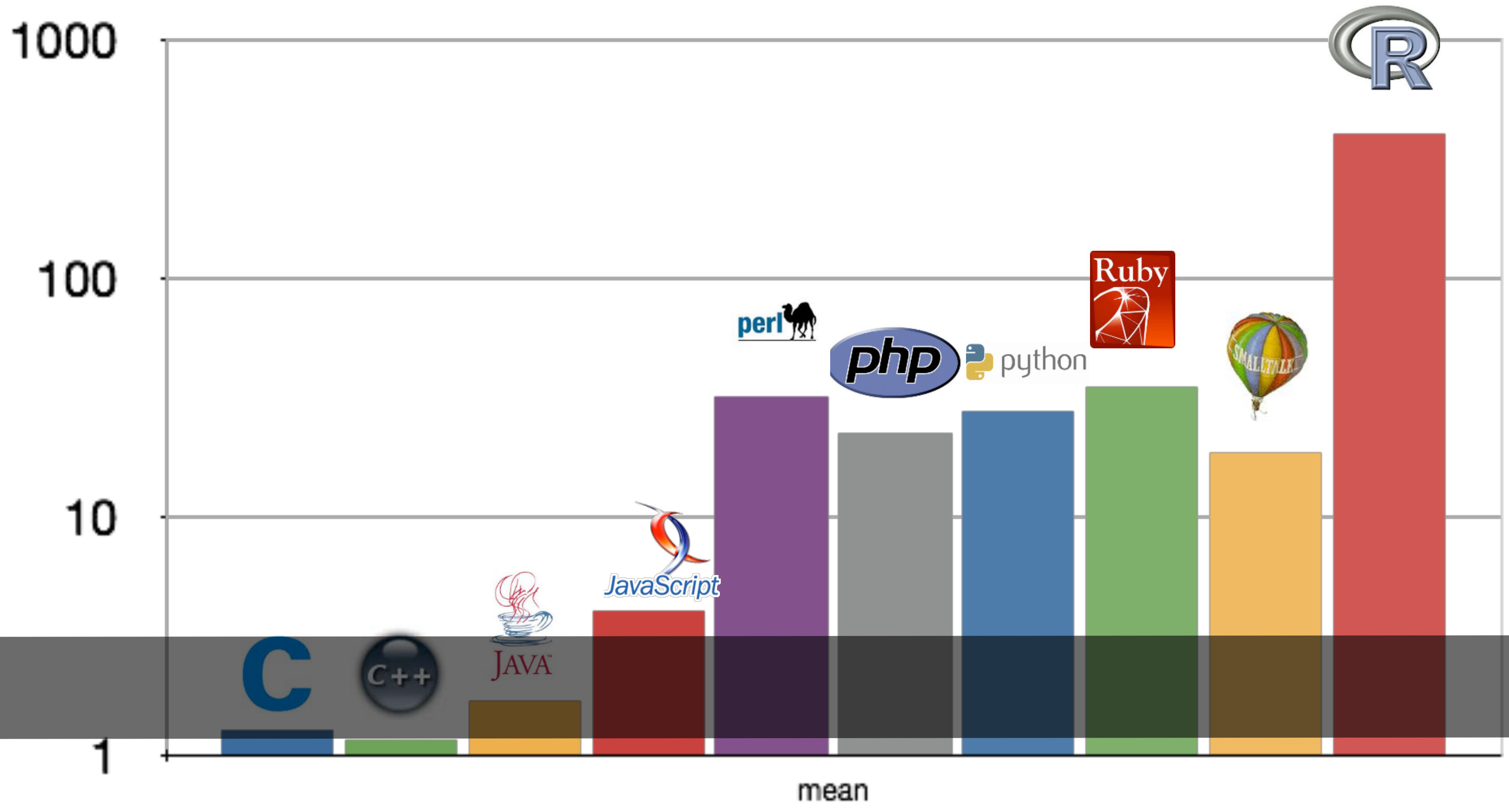
Why can't there be an "ultimate" programming language?

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 → **How it should be**

Current situation

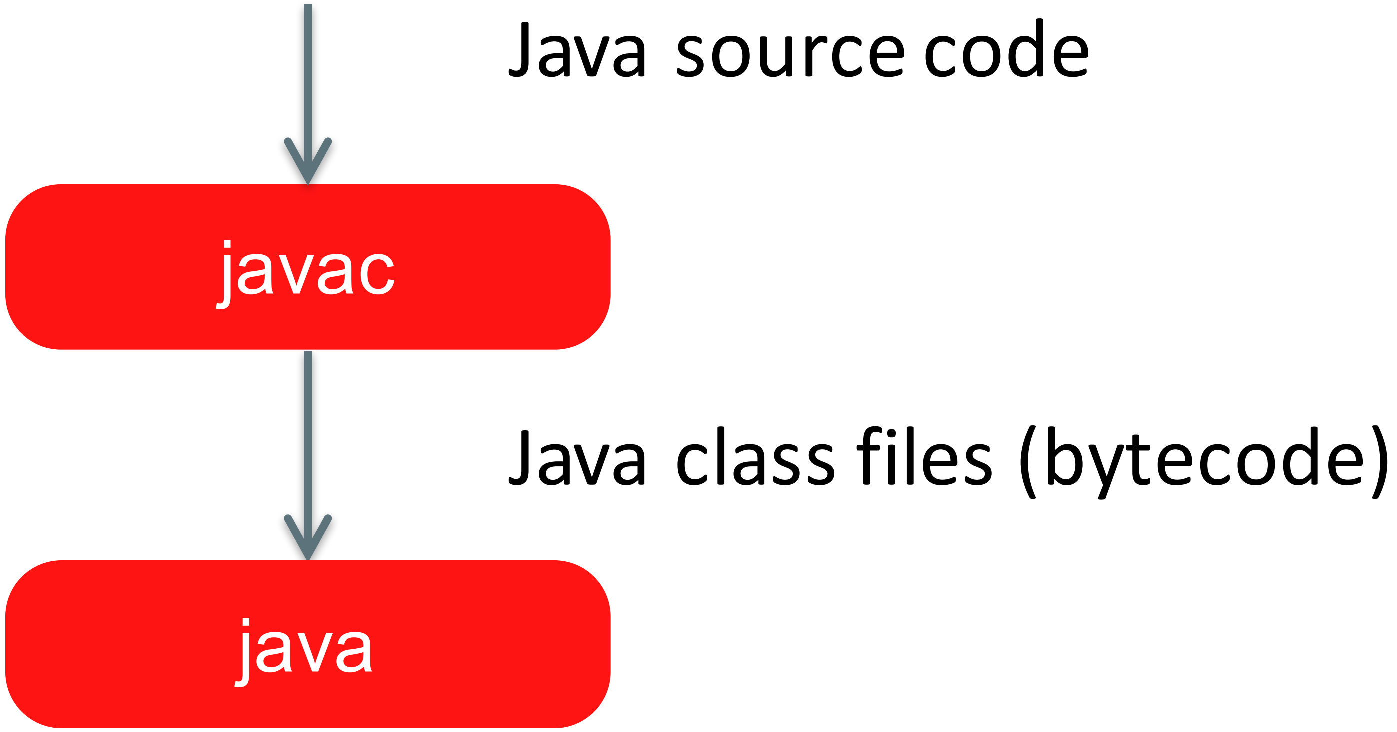
- **Prototype a new language**
  - Parser and language work to build syntax tree (AST), AST Interpreter
- **Write a "real" VM**
  - In C/C++, still using AST interpreter, spend a lot of time implementing runtime system, GC, …
- **People start using it**
- **People complain about performance**
  - Define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

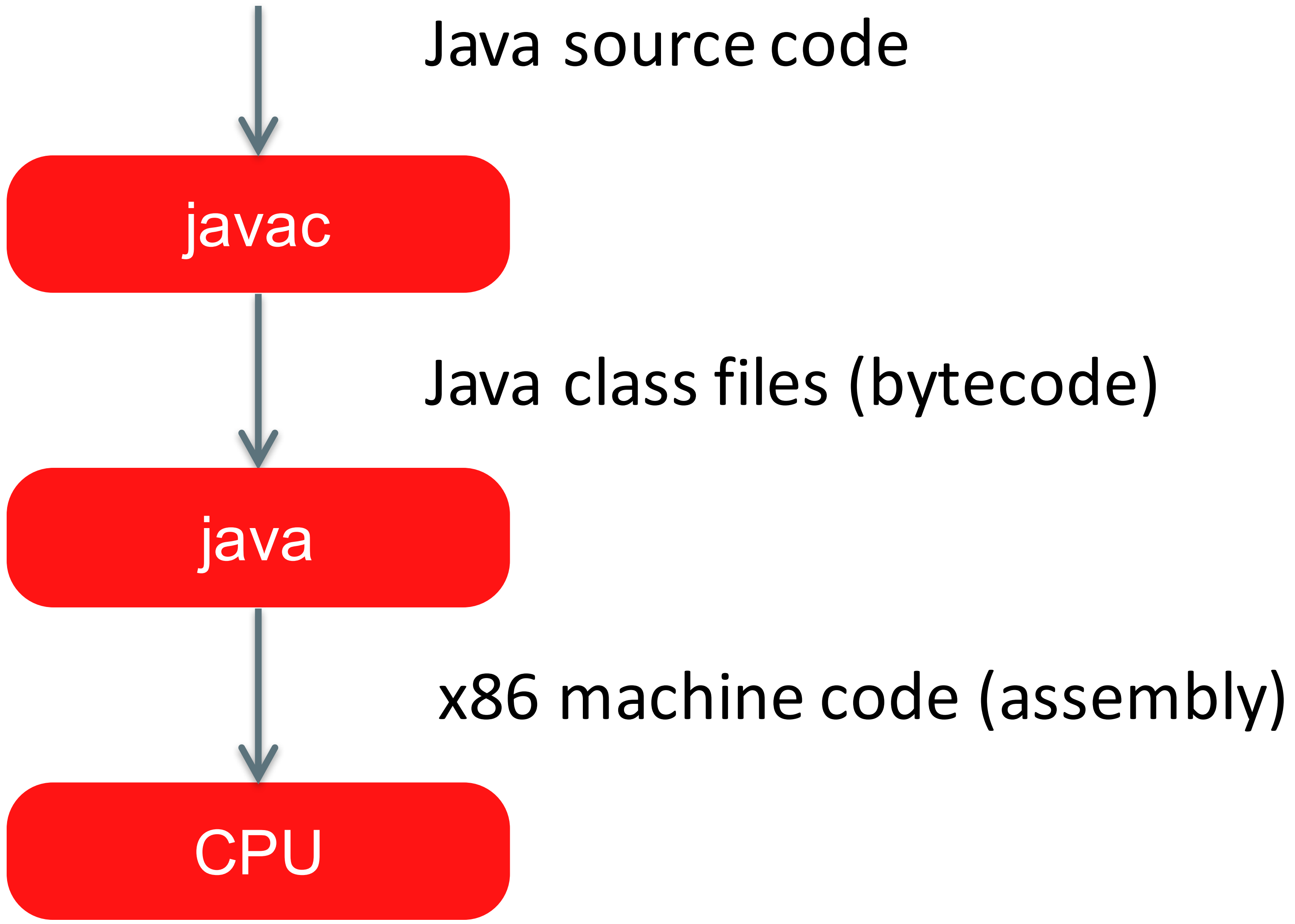
How it should be

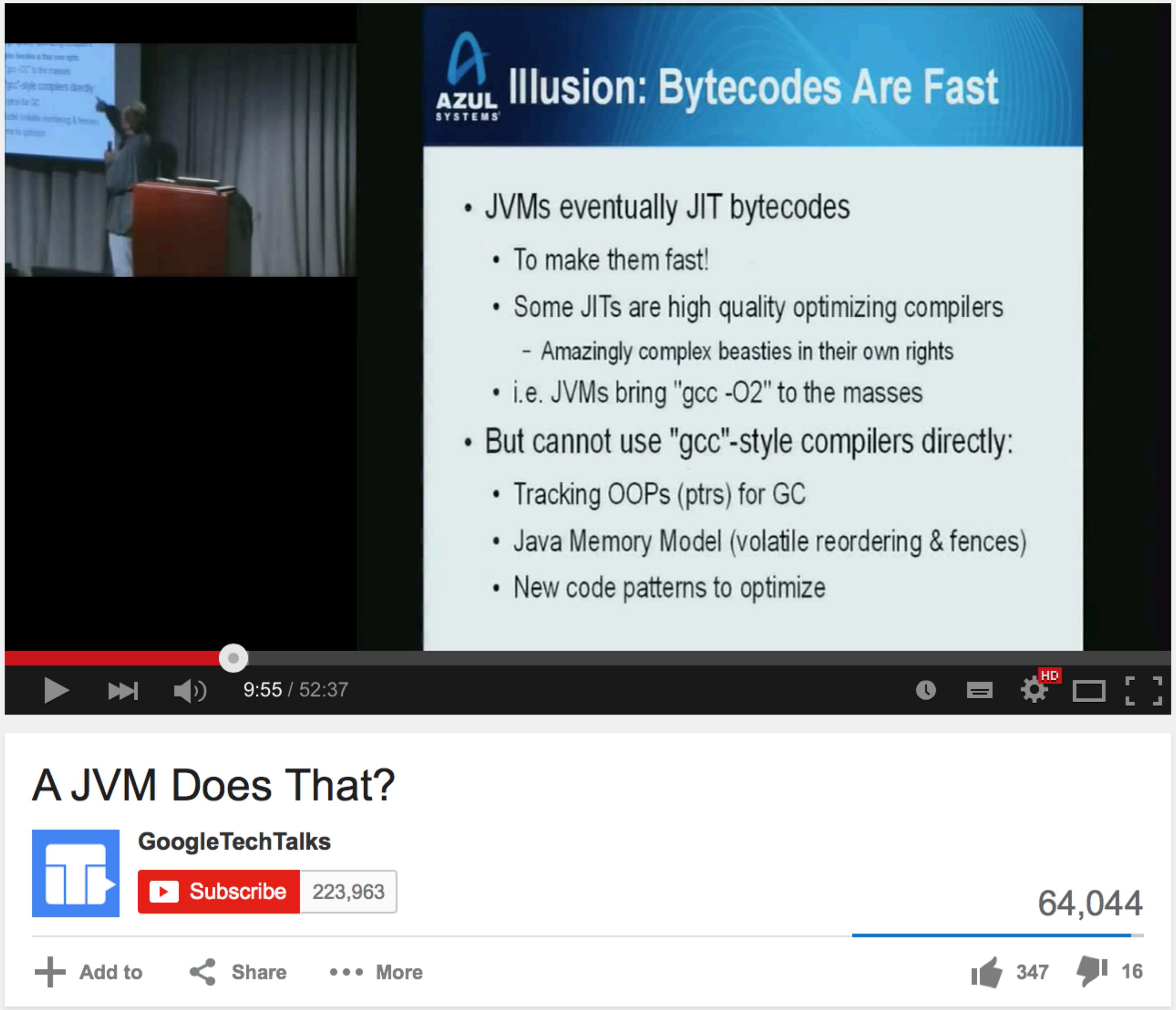
- **Prototype a new language in Java**
  - Parser and language work to build syntax tree (AST)
  - Execute using AST interpreter
- **People start using it**
  - And it is already fast

ORACLE

Java internals basics


Java source code
javac
Java class files (bytecode)
java


Java source code
javac
Java class files (bytecode)
java
x86 machine code (assembly)
CPU



Illusion: Bytecodes Are Fast

- JVMs eventually JIT bytecodes
  - To make them fast!
  - Some JITs are high quality optimizing compilers
    - Amazingly complex beasties in their own rights
  - i.e. JVMs bring "gcc -O2" to the masses
- But cannot use "gcc"-style compilers directly:
  - Tracking OOPs (ptrs) for GC
  - Java Memory Model (volatile reordering & fences)
  - New code patterns to optimize

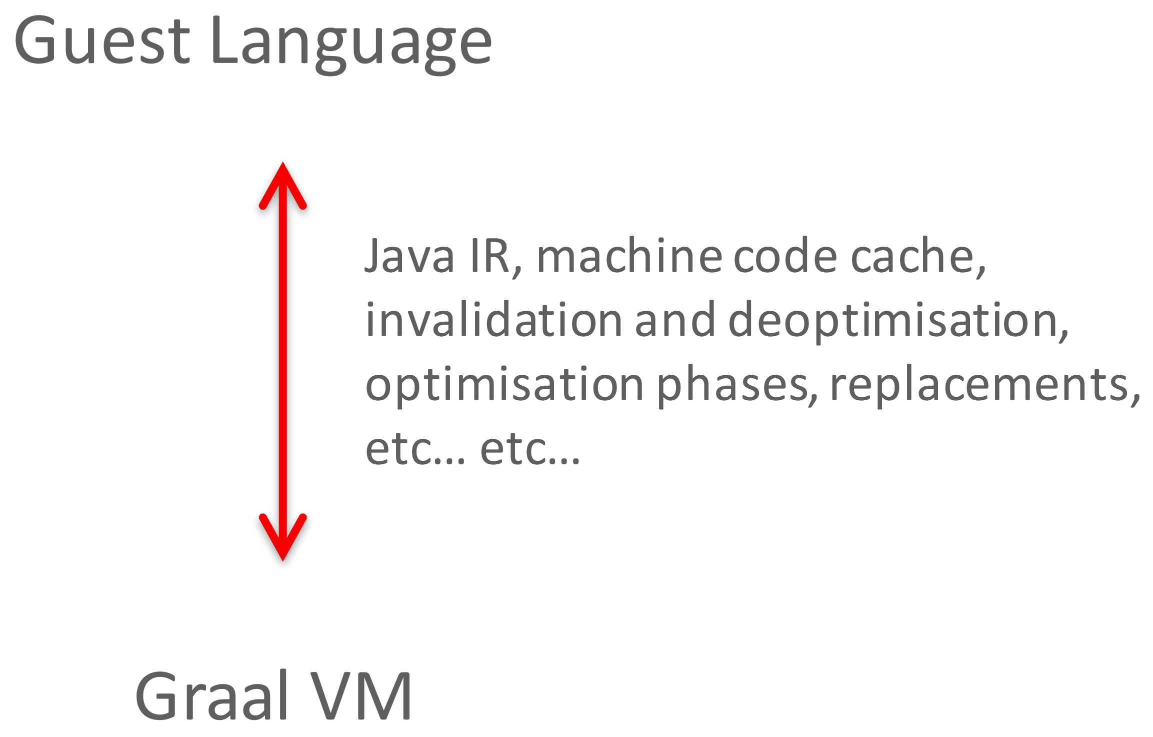
A JVM Does That?

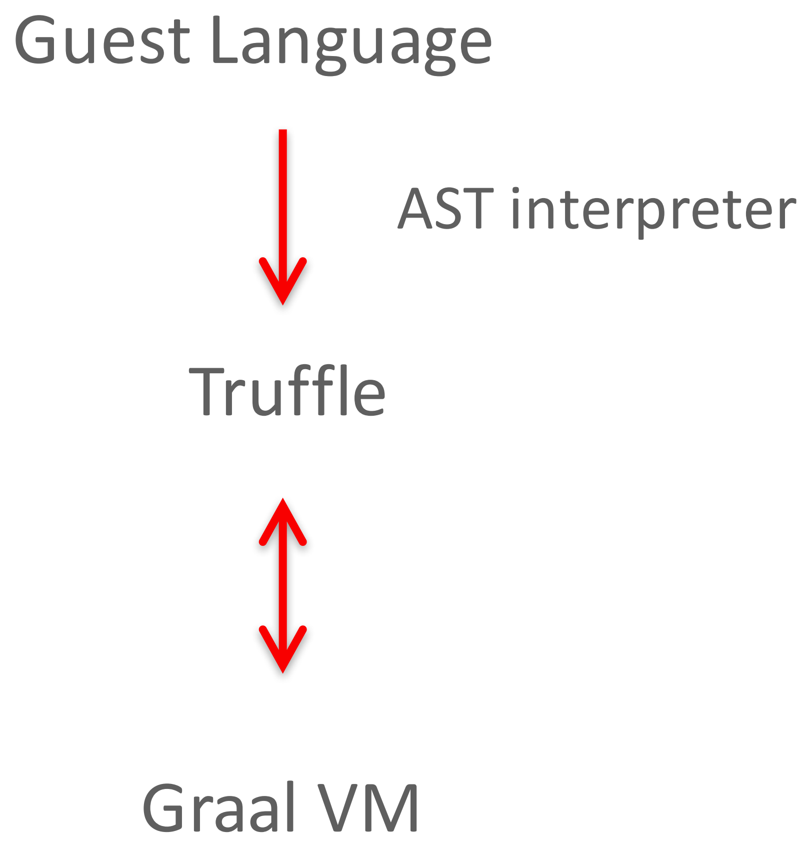
GoogleTechTalks

ORACLE

Truffle

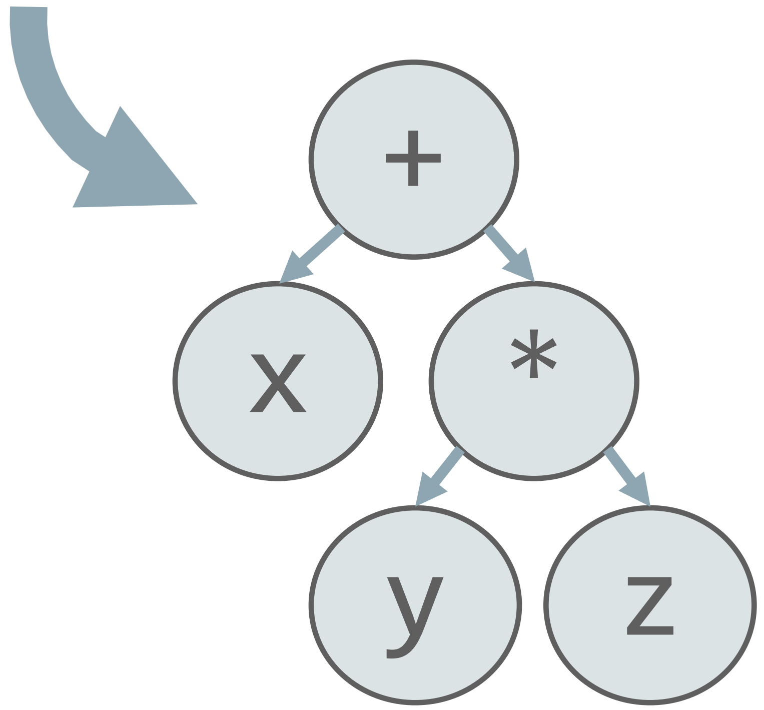
$$\begin{tikzcd}\text{Guest Language} \arrow[d, "\text{Bytecode}"] \\ \text{JVM}\end{tikzcd}$$


Guest Language
Java IR, machine code cache, invalidation and deoptimisation, optimisation phases, replacements, etc… etc…
Graal VM

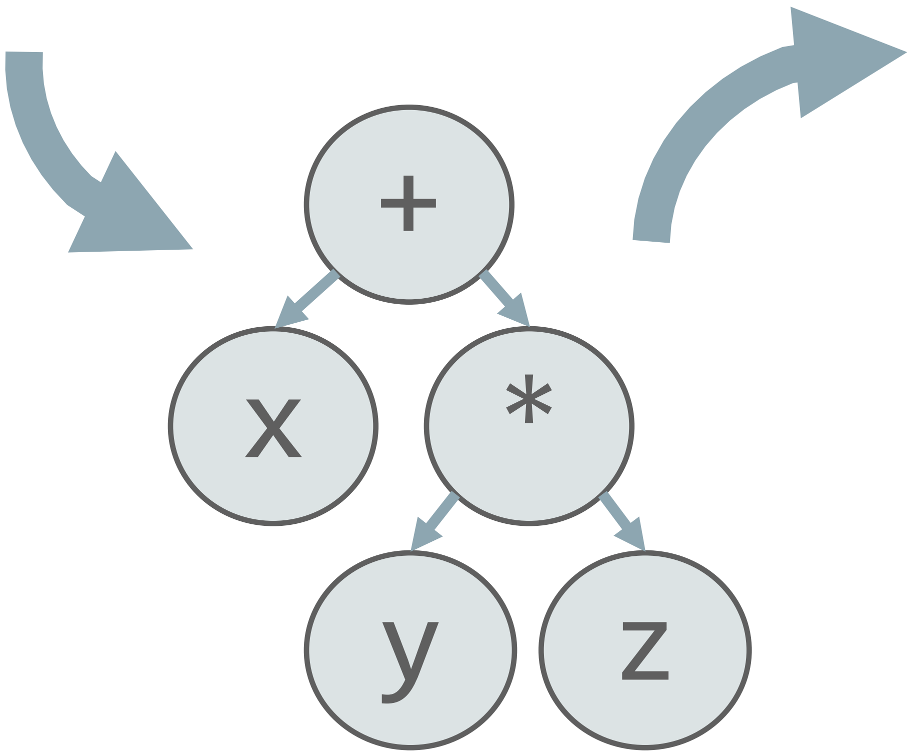

Guest Language
AST interpreter
Truffle
Graal VM

```
x + y * z
```

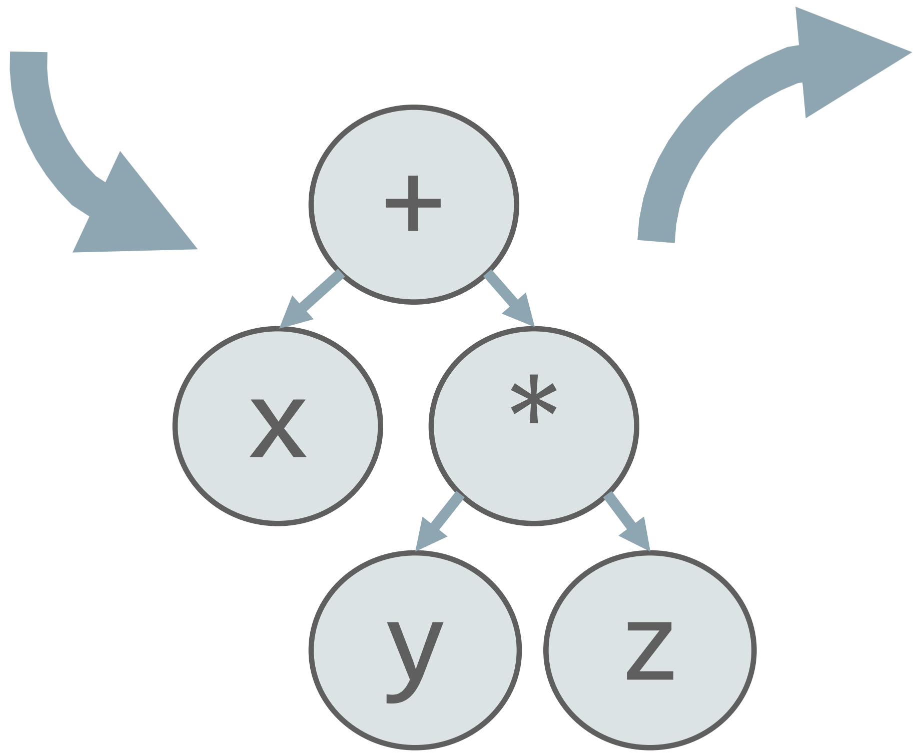
x + y * z



```
x + y * z
```



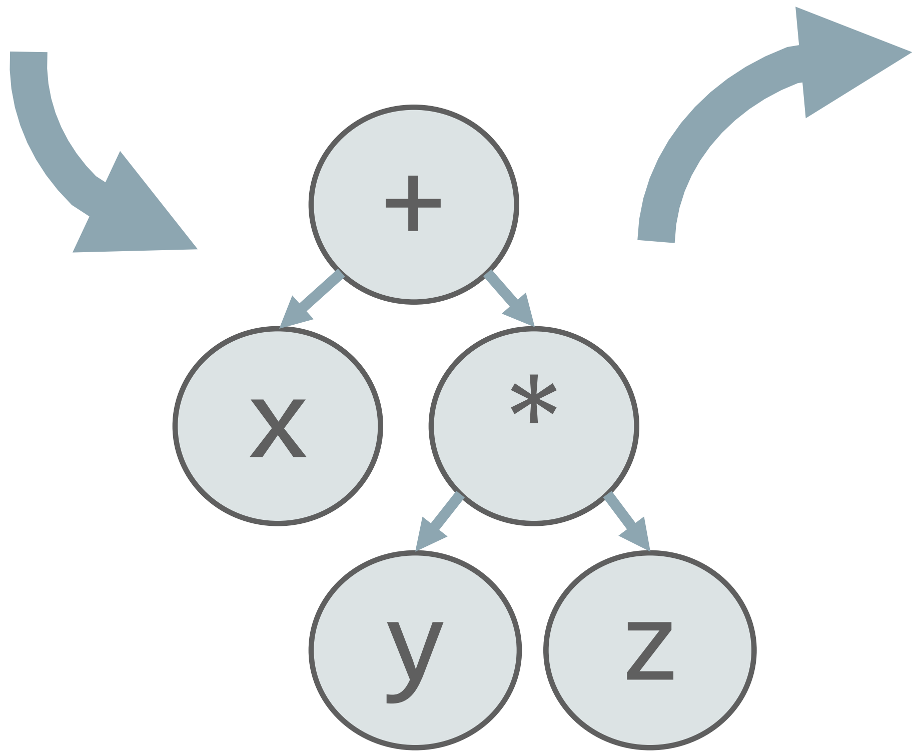
```
load_local x
load_local y
load_local z
call :*
call :+
x + y * z
```



```
load_local x
load_local y
load_local z
call :*
call :+
```

```
pushq %rbp
movq  %rsp, %rbp
movq  %rdi, -8(%rbp)
movq  %rsi, -16(%rbp)
movq  %rdx, -24(%rbp)
movq  -16(%rbp), %rax
movl  %eax, %edx
movq  -24(%rbp), %rax
imull %edx, %eax
movq  -8(%rbp), %rdx
addl  %edx, %eax
popq  %rbp
ret
```

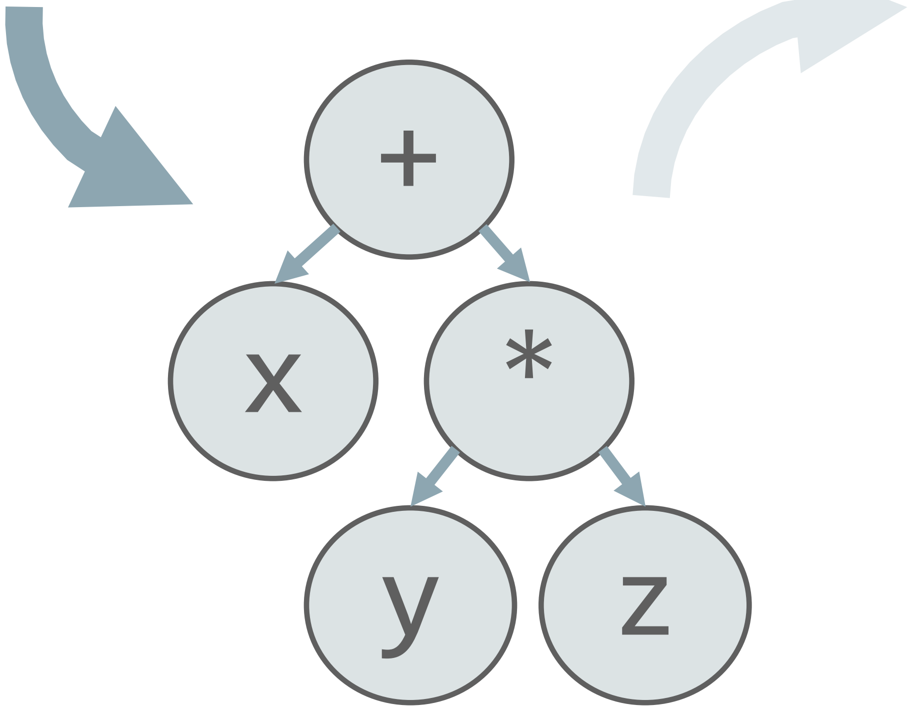
```
x + y * z
```



```
load_local x
load_local y
load_local z
call :*
call :+
```

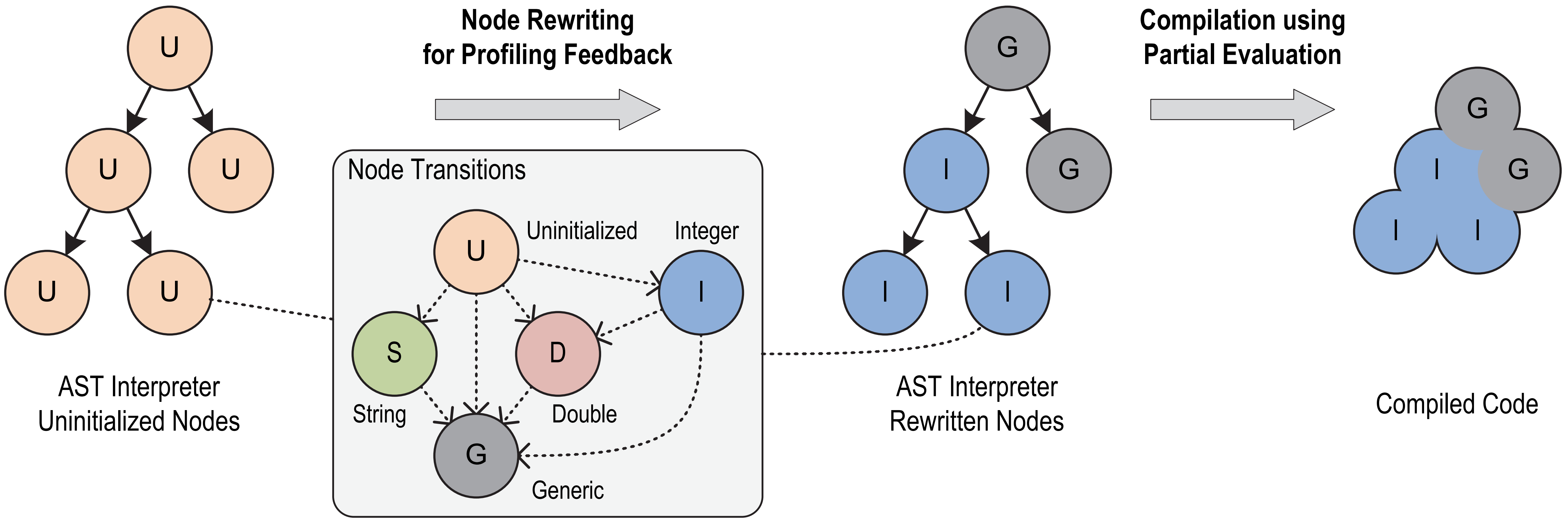
```
pushq %rbp
movq  %rsp, %rbp
movq  %rdi, -8(%rbp)
movq  %rsi, -16(%rbp)
movq  %rdx, -24(%rbp)
movq  -16(%rbp), %rax
movl  %eax, %edx
movq  -24(%rbp), %rax
imull %edx, %eax
movq  -8(%rbp), %rdx
addl  %edx, %eax
popq  %rbp
ret
```

```
x + y * z
```

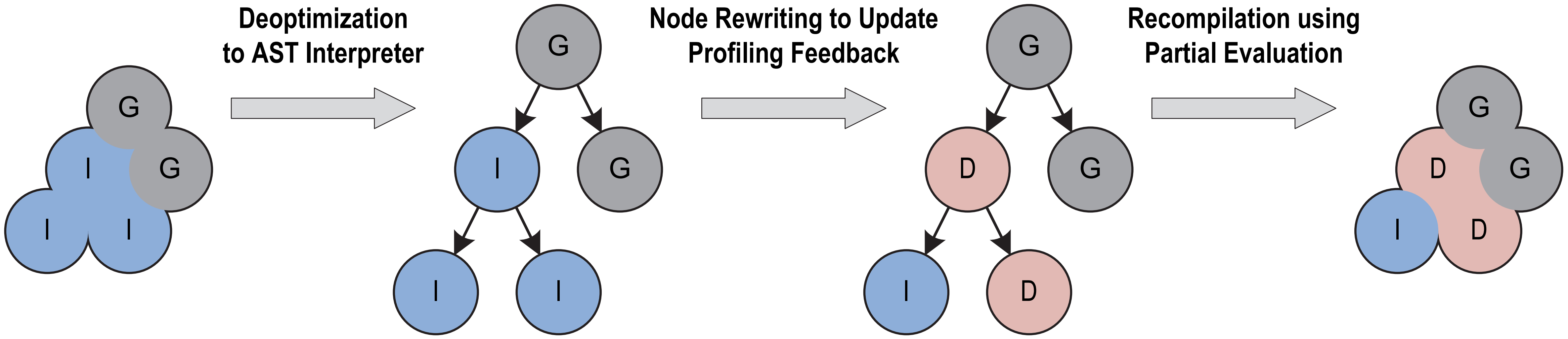


```
load_local x
load_local y
load_local z
call :*
call :+
```

```
pushq %rbp
movq  %rsp, %rbp
movq  %rdi, -8(%rbp)
movq  %rsi, -16(%rbp)
movq  %rdx, -24(%rbp)
movq  -16(%rbp), %rax
movl  %eax, %edx
movq  -24(%rbp), %rax
imull %edx, %eax
movq  -8(%rbp), %rdx
addl  %edx, %eax
popq  %rbp
ret
```



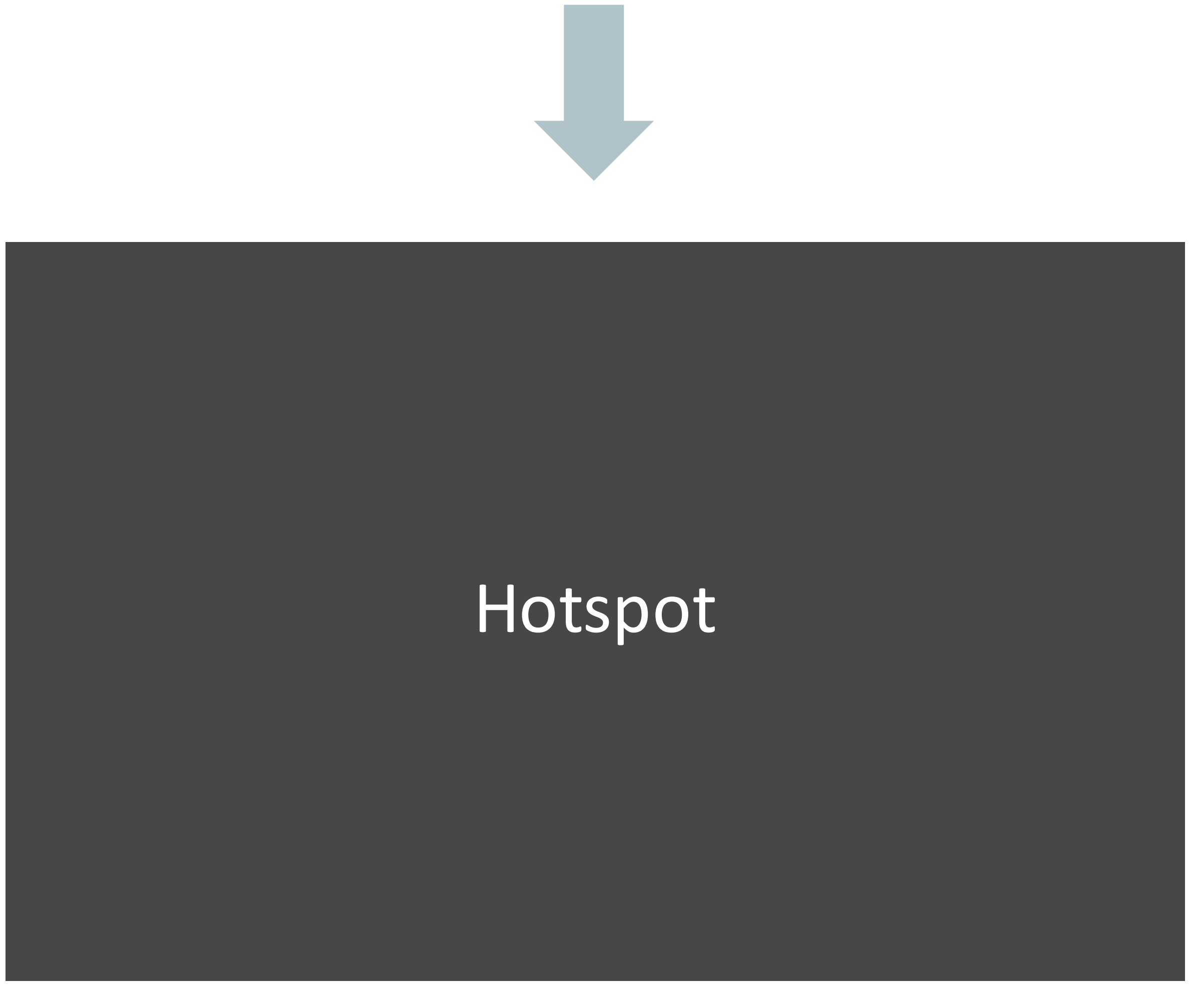
T. Würthinger, C. Wimmer, A. Wöß, L. Stadler, G. Duboscq, C. Humer, G. Richards, D. Simon, and M. Wolczko. One VM to rule them all. In Proceedings of Onward!, 2013.

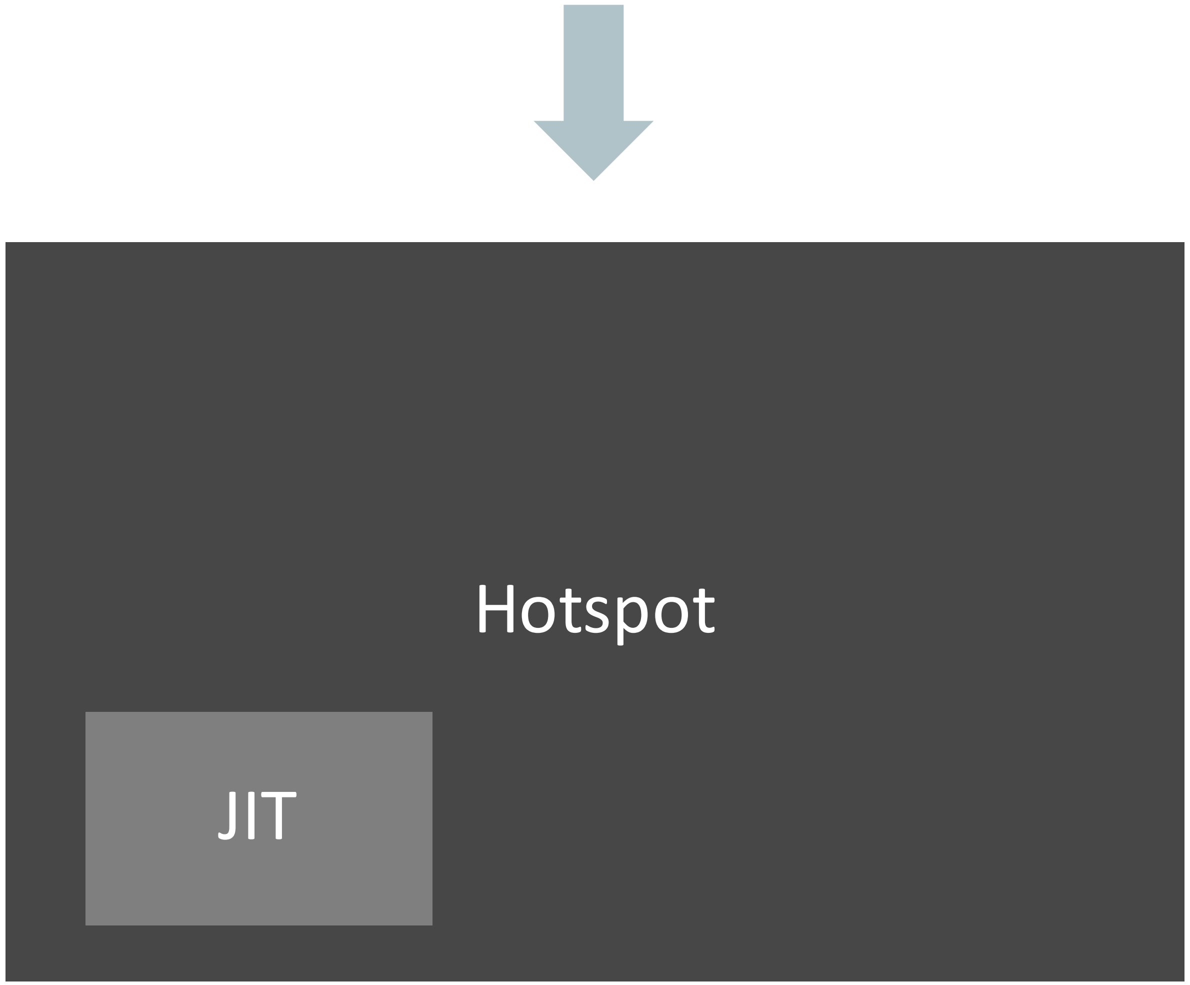

G
I
G
I
I
Deoptimization to AST Interpreter
G
I
G
I
I
Node Rewriting to Update Profiling Feedback
G
D
G
I
D
Recompilation using Partial Evaluation
G
D
G
I
D

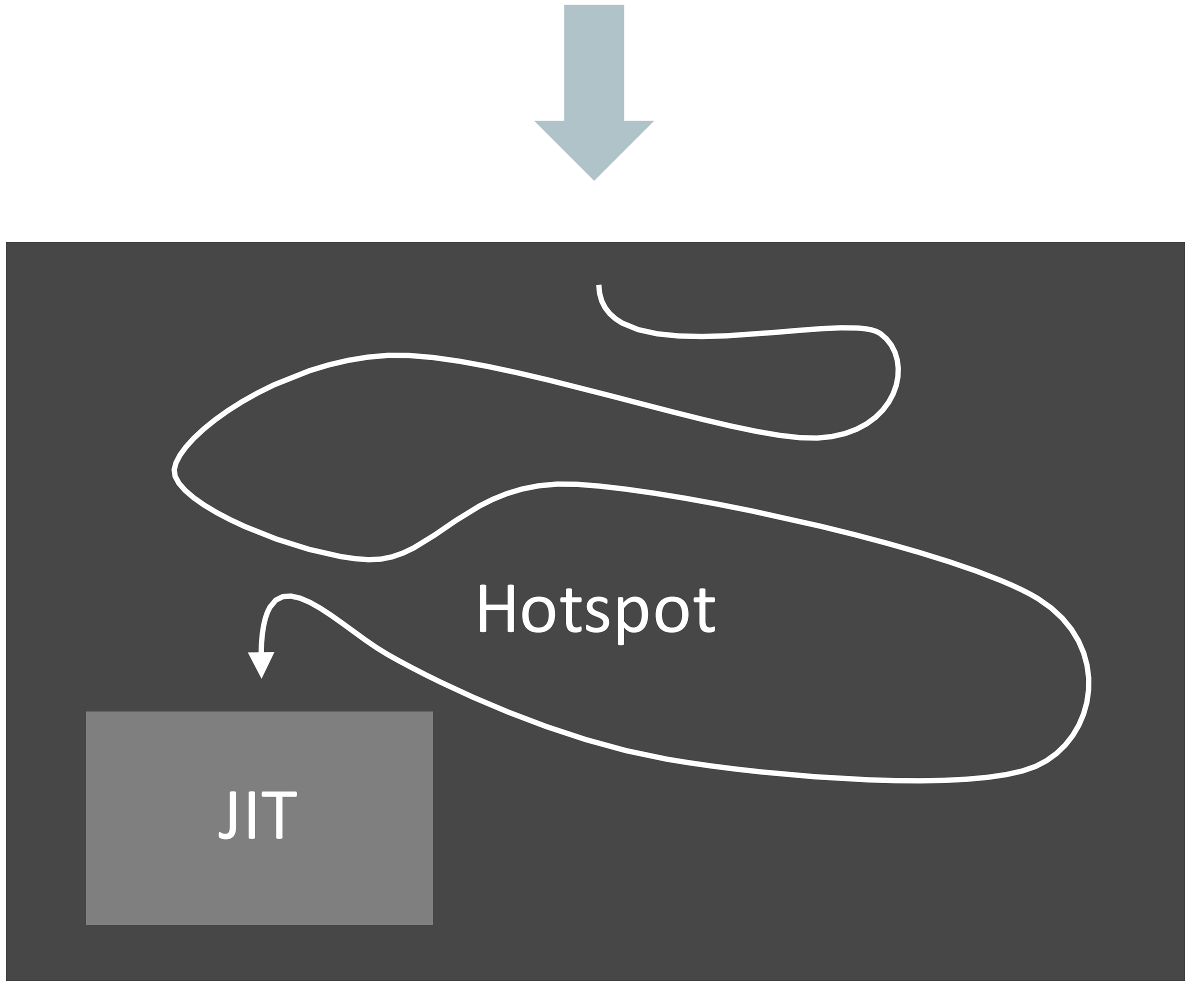
T. Würthinger, C. Wimmer, A. Wöß, L. Stadler, G. Duboscq, C. Humer, G. Richards, D. Simon, and M. Wolczko. One VM to rule them all. In Proceedings of Onward!, 2013.

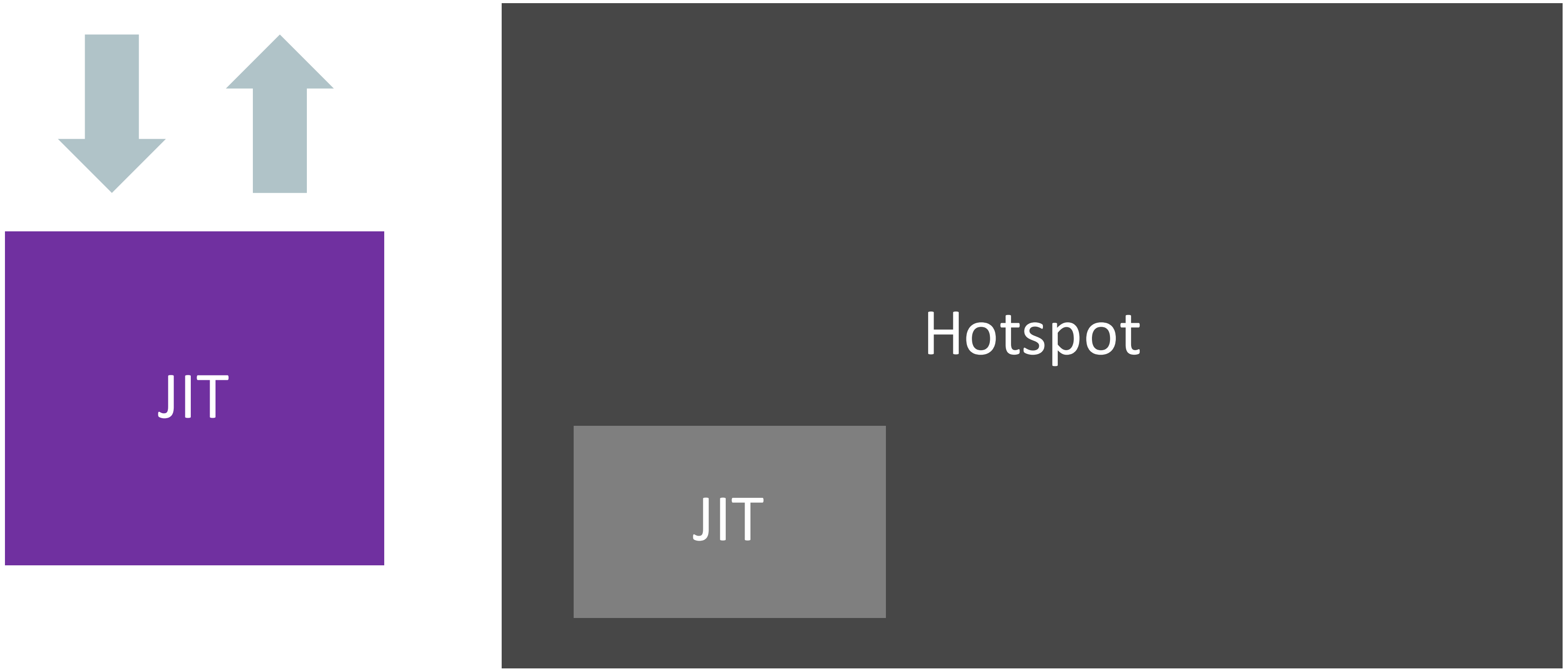
Graal

Hotspot

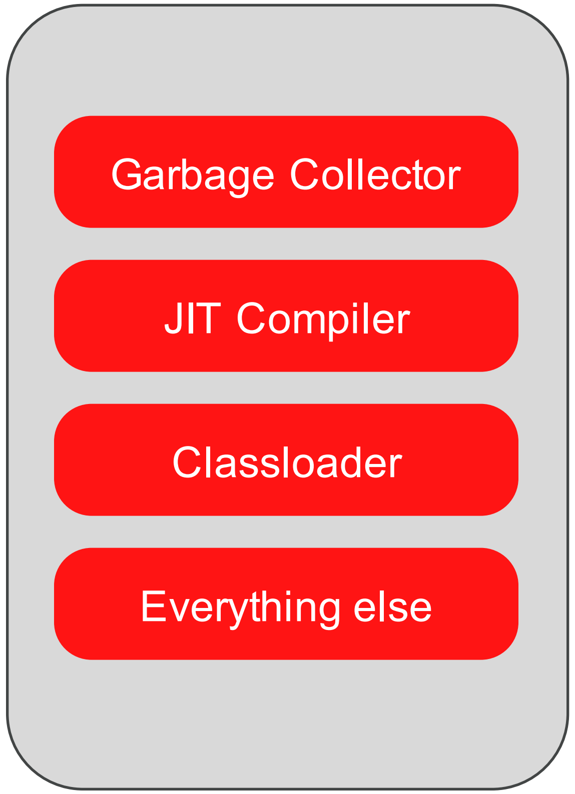

Hotspot


Hotspot
JIT

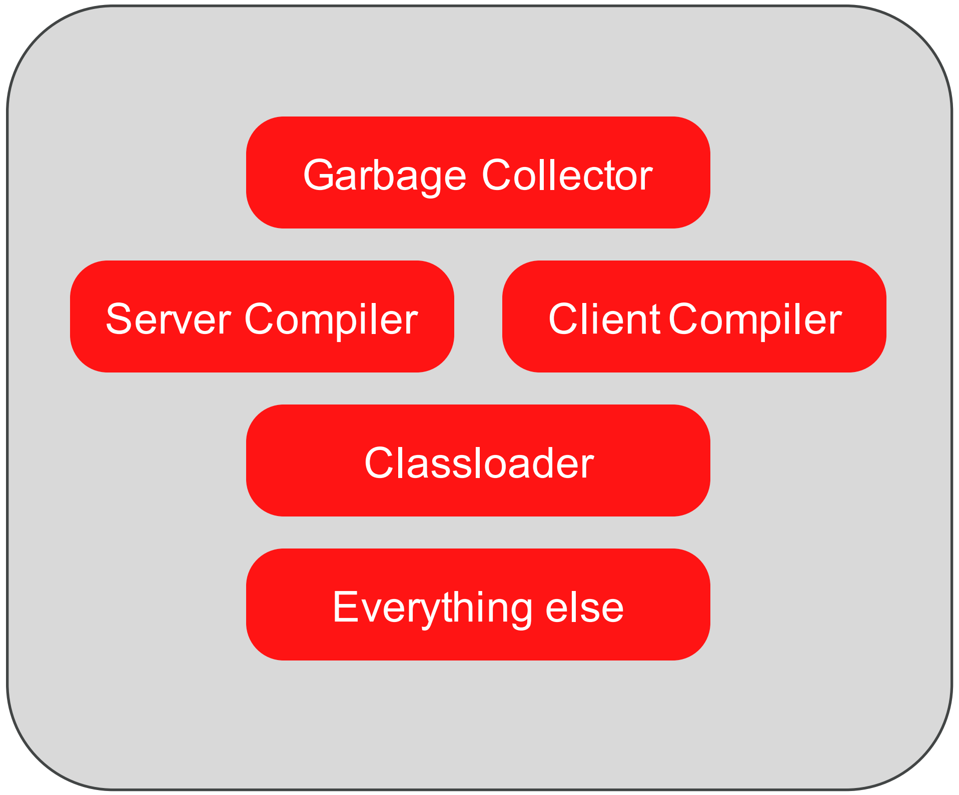

Hotspot
JIT


JIT
Hotspot
JIT

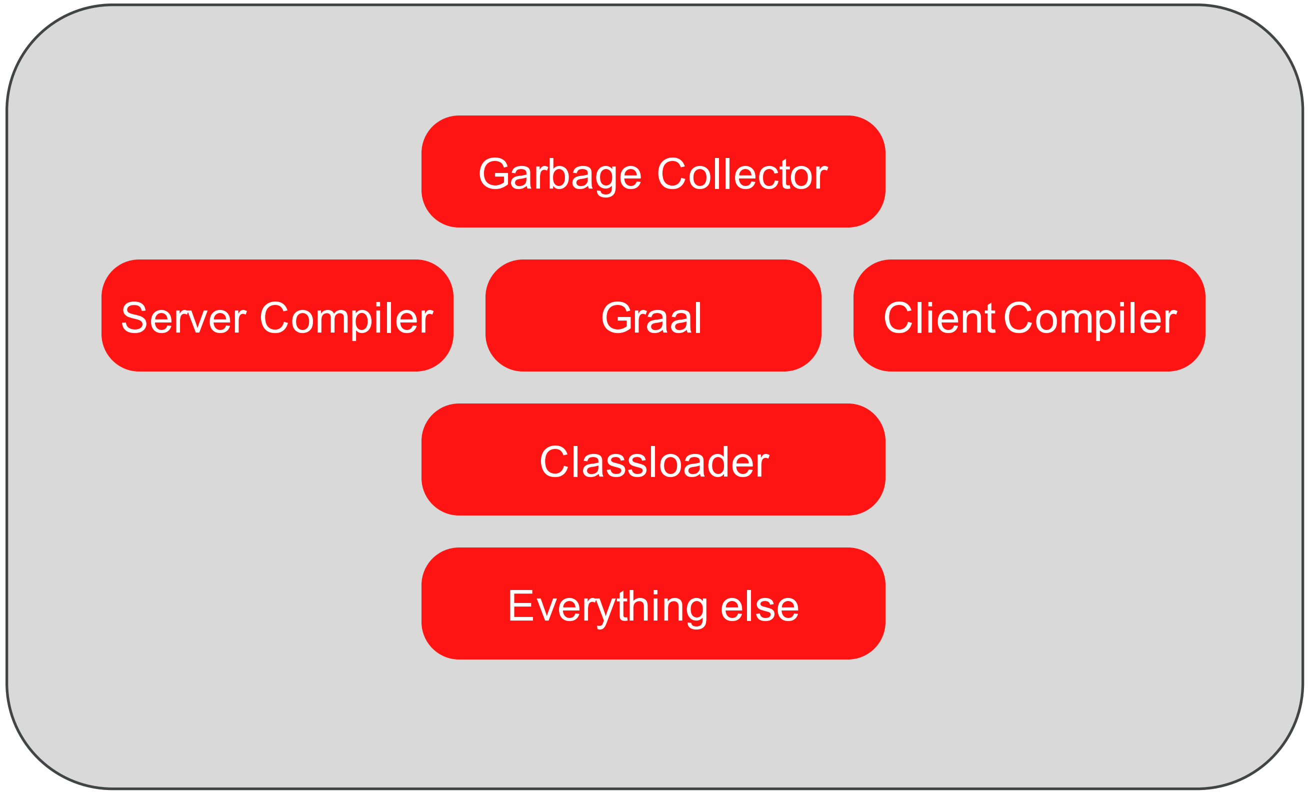
JVM

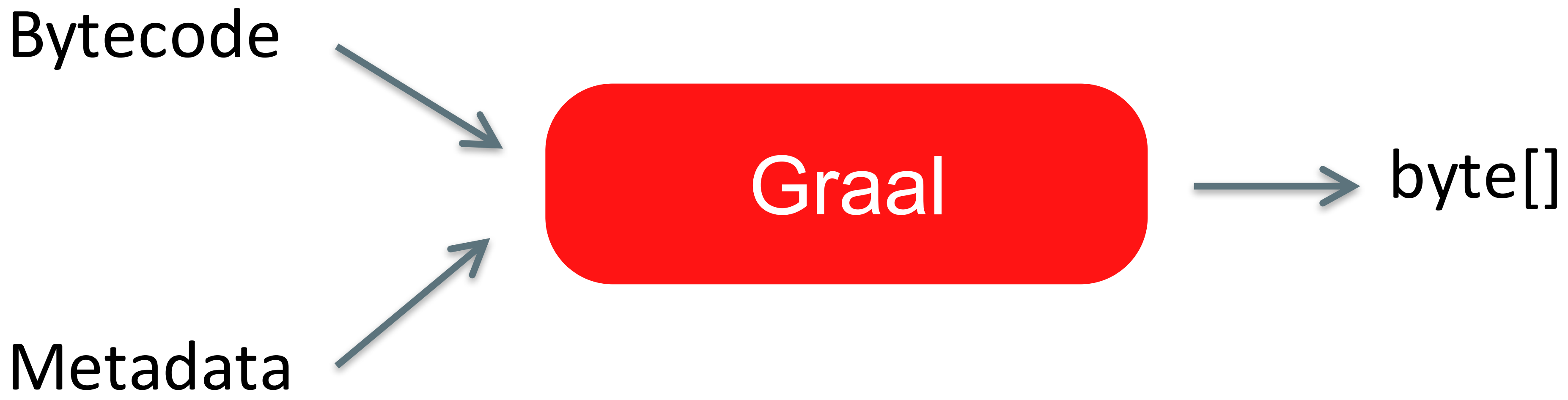

Garbage Collector
JIT Compiler
Classloader
Everything else

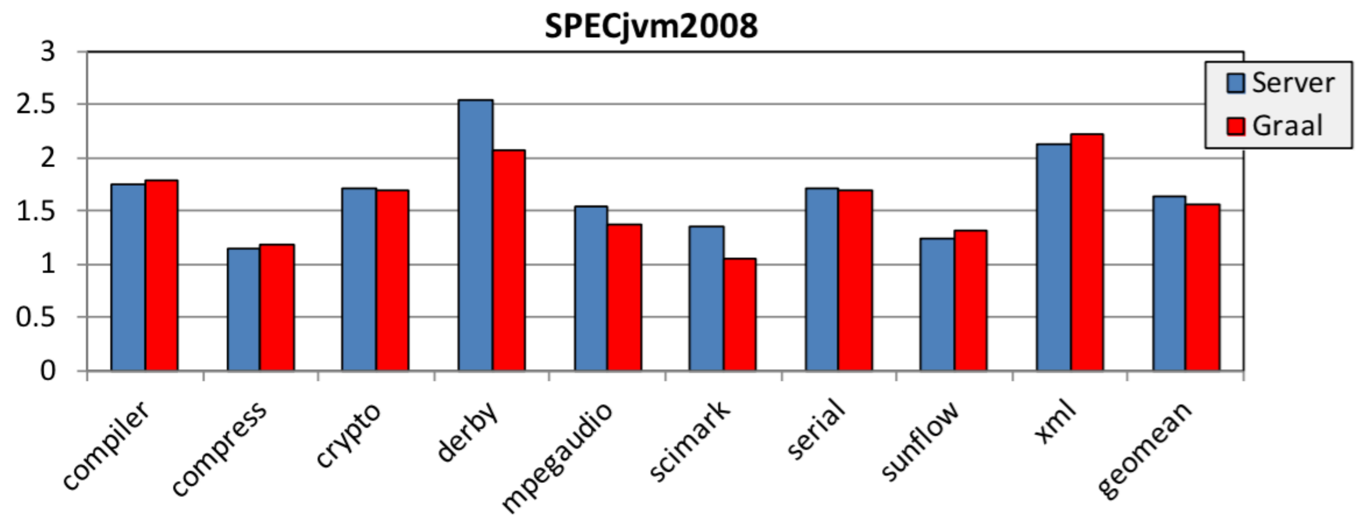
JVM

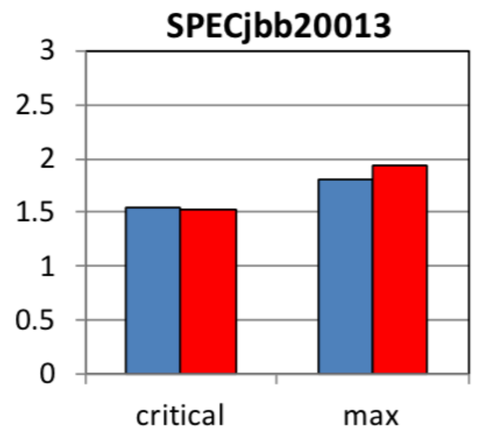

Garbage Collector
Server Compiler
Client Compiler
Classloader
Everything else

JVM


Garbage Collector
Server Compiler
Graal
Client Compiler
Classloader
Everything else

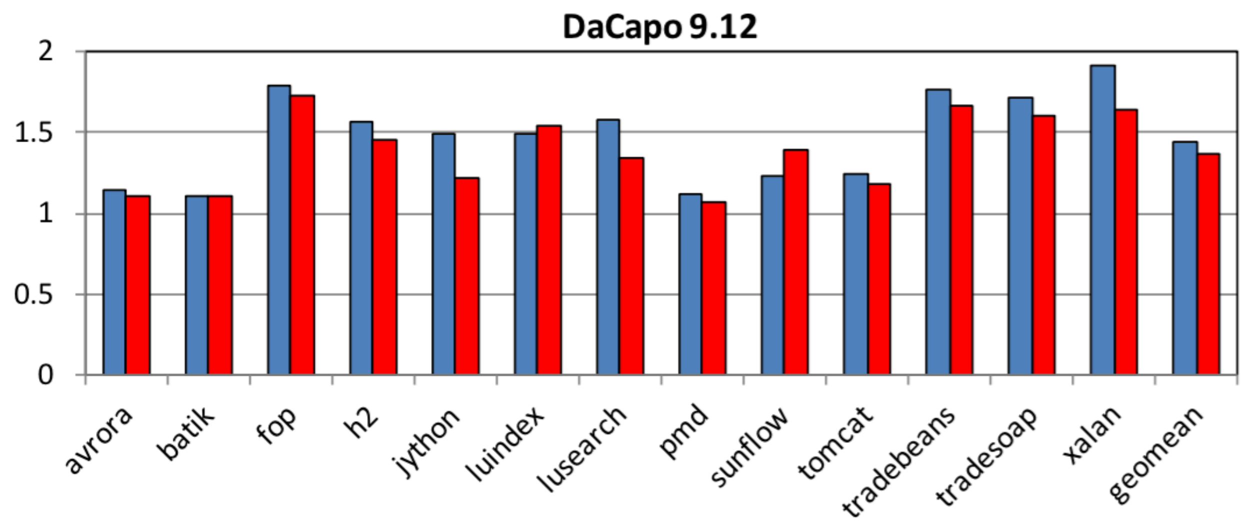

Bytecode
Metadata
Graal
byte[]

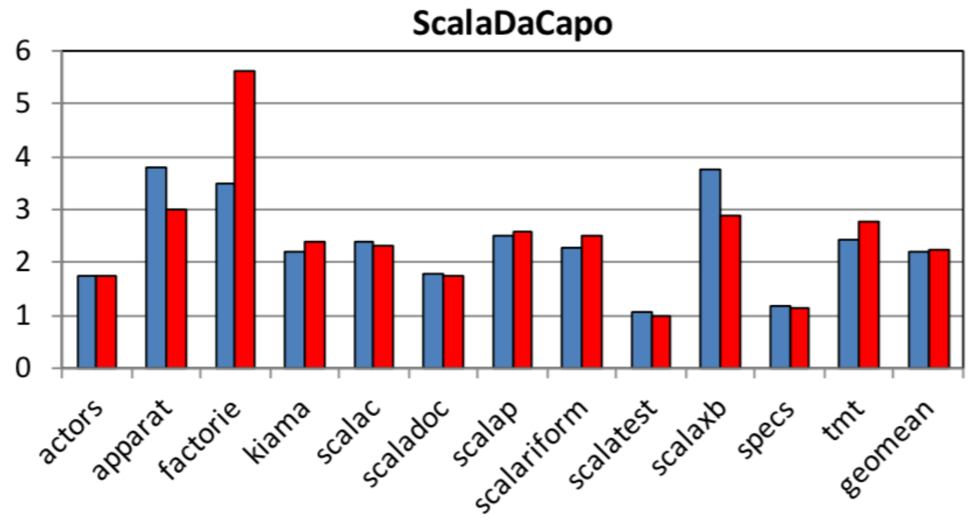




Higher is better, normalized to Client compiler.

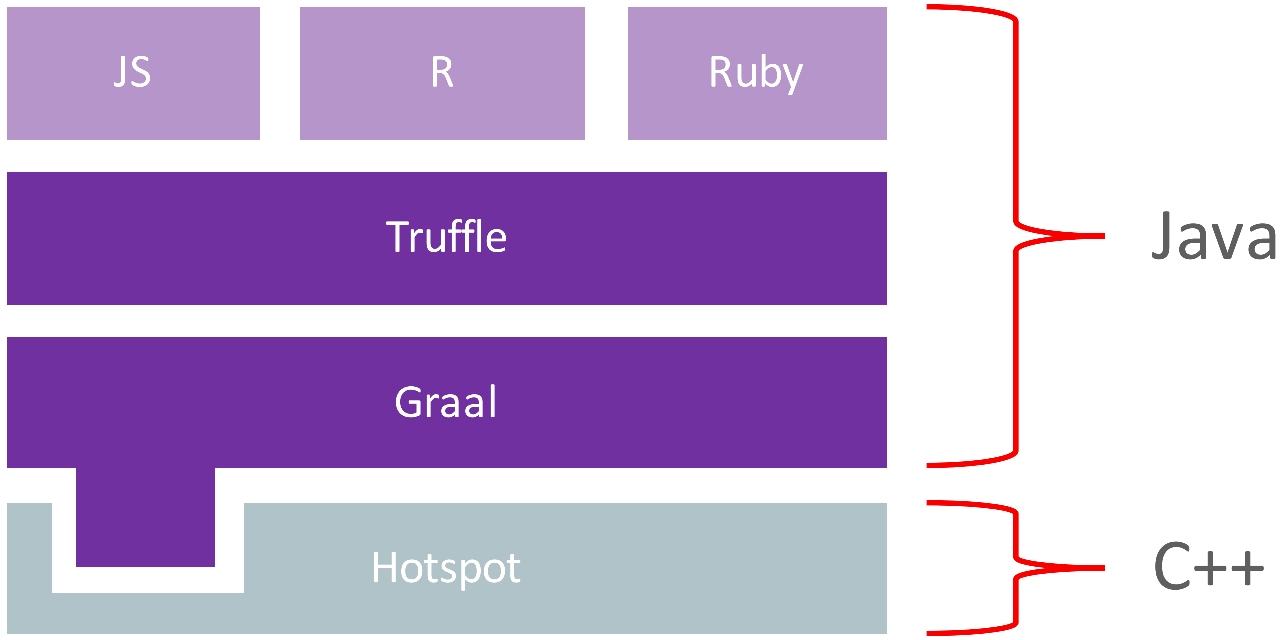
Results are not SPEC compliant, but follow the rules for research use.



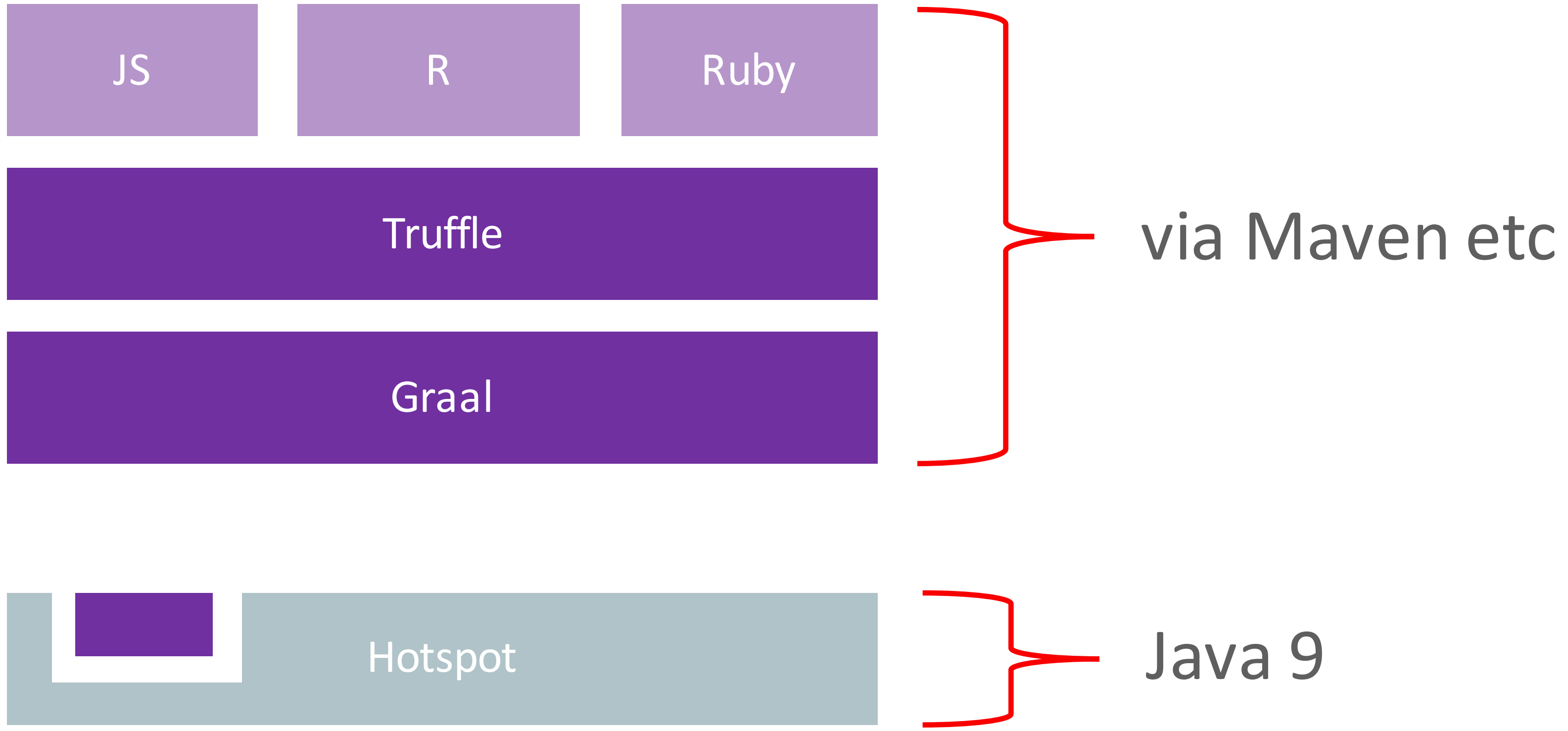


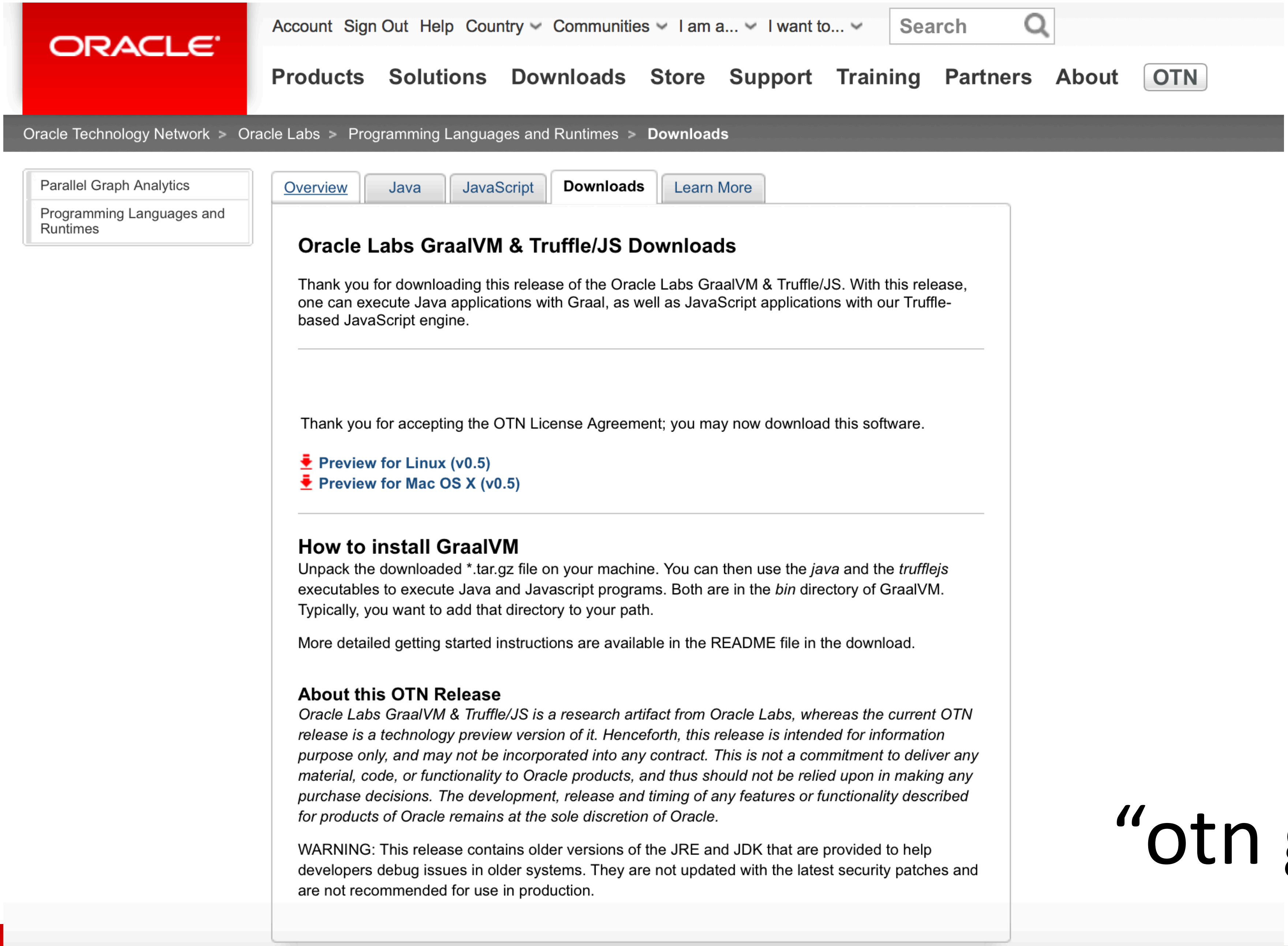
C. Wimmer, Graal Tutorial, 2015.

JVMCI
(JVM Compiler Interface)



ORACLE


JS
R
Ruby
Truffle
Graal
Hotspot
via Maven etc
Java 9


ORACLE
Account Sign Out Help Country Communities I am a... I want to...
Search
Products Solutions Downloads Store Support Training Partners About OTN
Oracle Technology Network > Oracle Labs > Programming Languages and Runtimes > Downloads
Parallel Graph Analytics
Programming Languages and Runtimes
Overview Java JavaScript Downloads Learn More

Oracle Labs GraalVM & Truffle/JS Downloads

Thank you for downloading this release of the Oracle Labs GraalVM & Truffle/JS. With this release, one can execute Java applications with Graal, as well as JavaScript applications with our Truffle-based JavaScript engine.

Thank you for accepting the OTN License Agreement; you may now download this software.

Preview for Linux (v0.5)
Preview for Mac OS X (v0.5)

How to install GraalVM

Unpack the downloaded *.tar.gz file on your machine. You can then use the *java* and the *trufflejs* executables to execute Java and Javascript programs. Both are in the *bin* directory of GraalVM. Typically, you want to add that directory to your path.

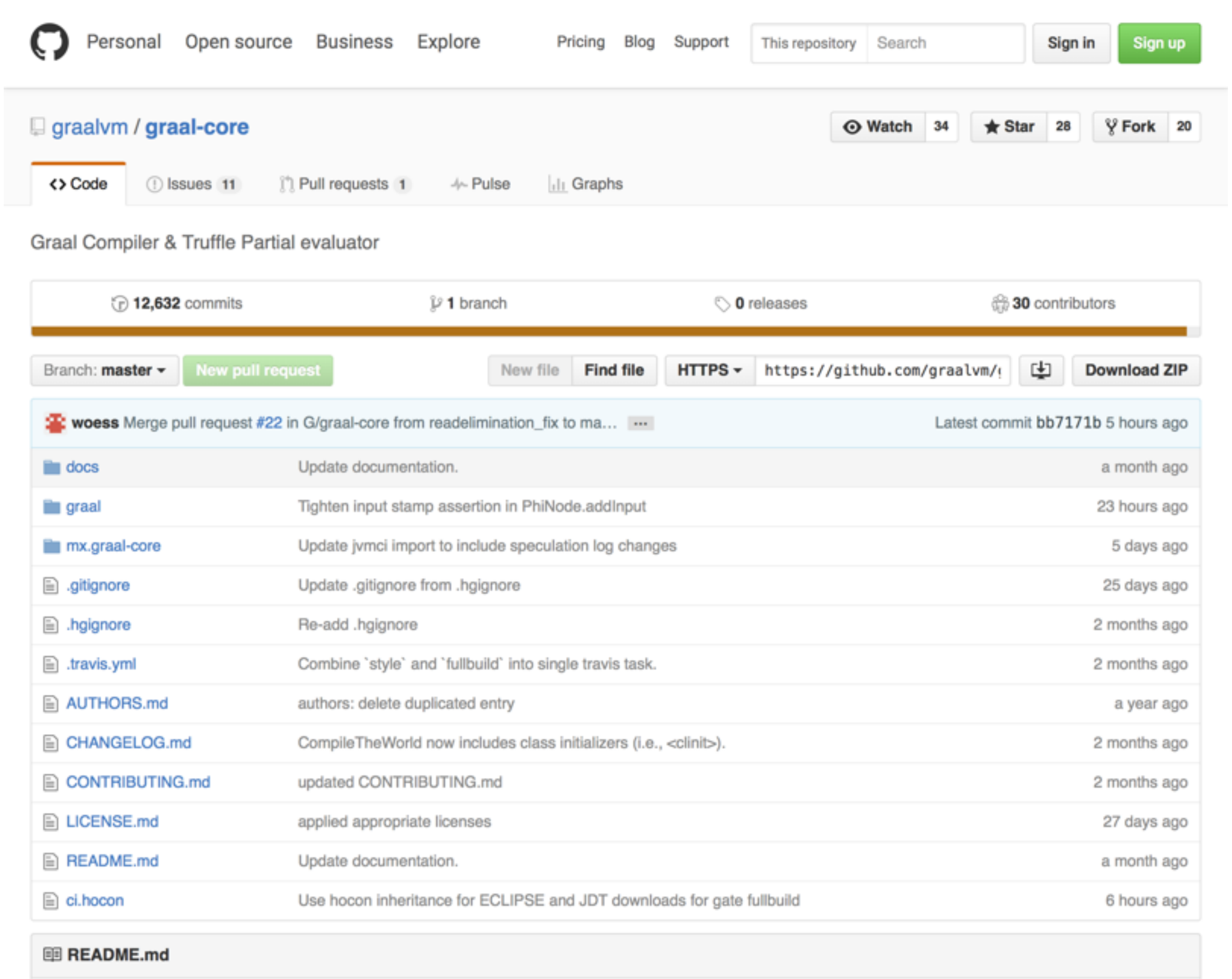
More detailed getting started instructions are available in the README file in the download.

About this OTN Release

Oracle Labs GraalVM & Truffle/JS is a research artifact from Oracle Labs, whereas the current OTN release is a technology preview version of it. Henceforth, this release is intended for information purpose only, and may not be incorporated into any contract. This is not a commitment to deliver any material, code, or functionality to Oracle products, and thus should not be relied upon in making any purchase decisions. The development, release and timing of any features or functionality described for products of Oracle remains at the sole discretion of Oracle.

WARNING: This release contains older versions of the JRE and JDK that are provided to help developers debug issues in older systems. They are not updated with the latest security patches and are not recommended for use in production.

“otn graal”


Personal Open source Business Explore Pricing Blog Support This repository Search Sign in Sign up

graalvm / graal-core

Watch 34 Star 28 Fork 20

Code Issues 11 Pull requests 1 Pulse Graphs

Graal Compiler & Truffle Partial evaluator

12,632 commits 1 branch 0 releases 30 contributors

Branch: master New pull request New file Find file HTTPS https://github.com/graalvm/ Download ZIP

woess Merge pull request #22 in G/graal-core from readelimination_fix to ma… Latest commit bb7171b 5 hours ago

docs	Update documentation.	a month ago
graal	Tighten input stamp assertion in PhiNode.addInput	23 hours ago
mx.graal-core	Update jvmci import to include speculation log changes	5 days ago
.gitignore	Update .gitignore from .hgignore	25 days ago
.hgignore	Re-add .hgignore	2 months ago
.travis.yml	Combine `style` and `fullbuild` into single travis task.	2 months ago
AUTHORS.md	authors: delete duplicated entry	a year ago
CHANGELOG.md	CompileTheWorld now includes class initializers (i.e., <clinit>).	2 months ago
CONTRIBUTING.md	updated CONTRIBUTING.md	2 months ago
LICENSE.md	applied appropriate licenses	27 days ago
README.md	Update documentation.	a month ago
ci.hocon	Use hocon inheritance for ECLIPSE and JDT downloads for gate fullbuild	6 hours ago

README.md

Ruby

JRuby

JITs by emitting JVM bytecode
VM in Java
Core library mostly in Java

Rubinius

JITs by emitting LLVM code
VM in C++
Core library mostly in Ruby

+ Truffle and Graal

100%

Compatibility with the language (spec/ruby)

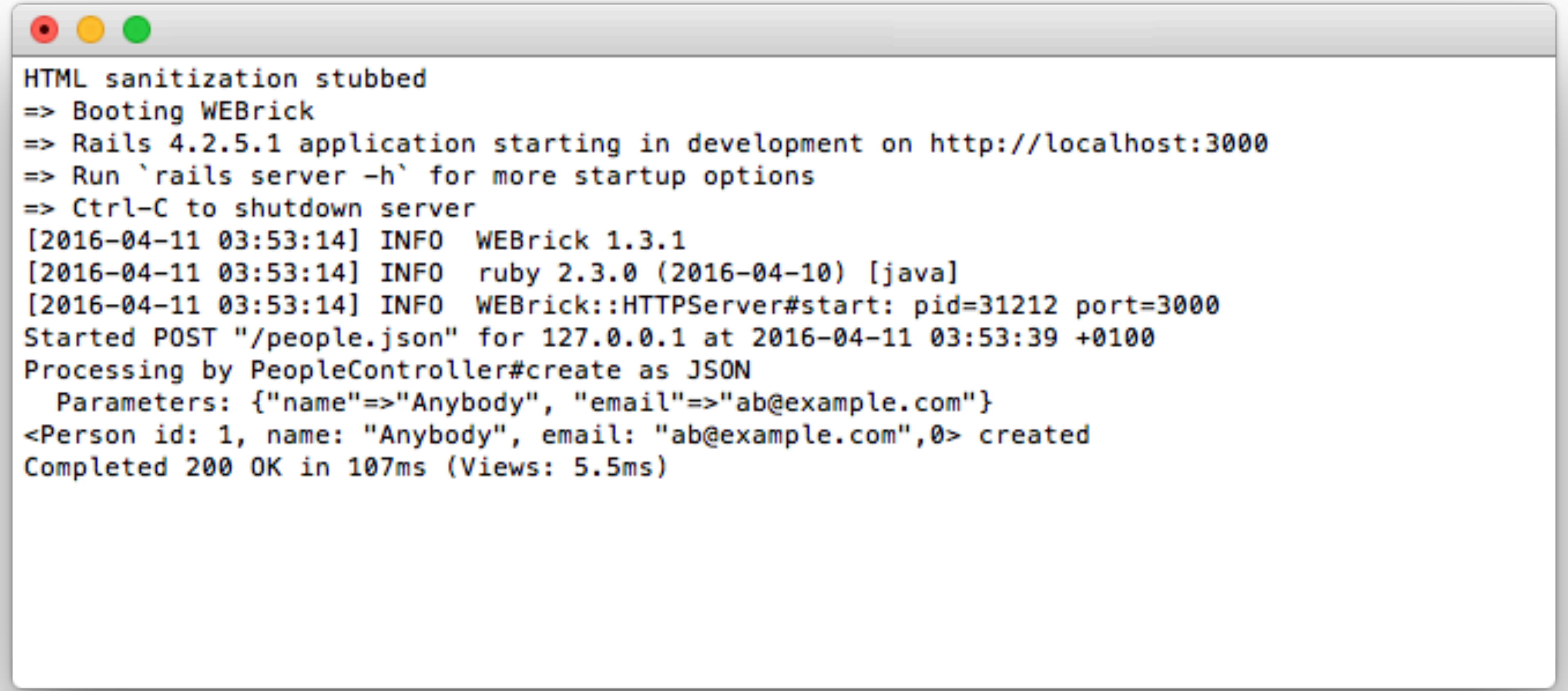
ORACLE

90%

Compatibility with the core library (spec/ruby)

ORACLE®

But does it run Rails?



```
HTML sanitization stubbed
=> Booting WEBrick
=> Rails 4.2.5.1 application starting in development on http://localhost:3000
=> Run `rails server -h` for more startup options
=> Ctrl-C to shutdown server
[2016-04-11 03:53:14] INFO  WEBrick 1.3.1
[2016-04-11 03:53:14] INFO  ruby 2.3.0 (2016-04-10) [java]
[2016-04-11 03:53:14] INFO  WEBrick::HTTPServer#start: pid=31212 port=3000
Started POST "/people.json" for 127.0.0.1 at 2016-04-11 03:53:39 +0100
Processing by PeopleController#create as JSON
  Parameters: {"name"=>"Anybody", "email"=>"ab@example.com"}
<Person id: 1, name: "Anybody", email: "ab@example.com",0> created
Completed 200 OK in 107ms (Views: 5.5ms)
```

Why is it apparently so hard to make Ruby fast?

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
```

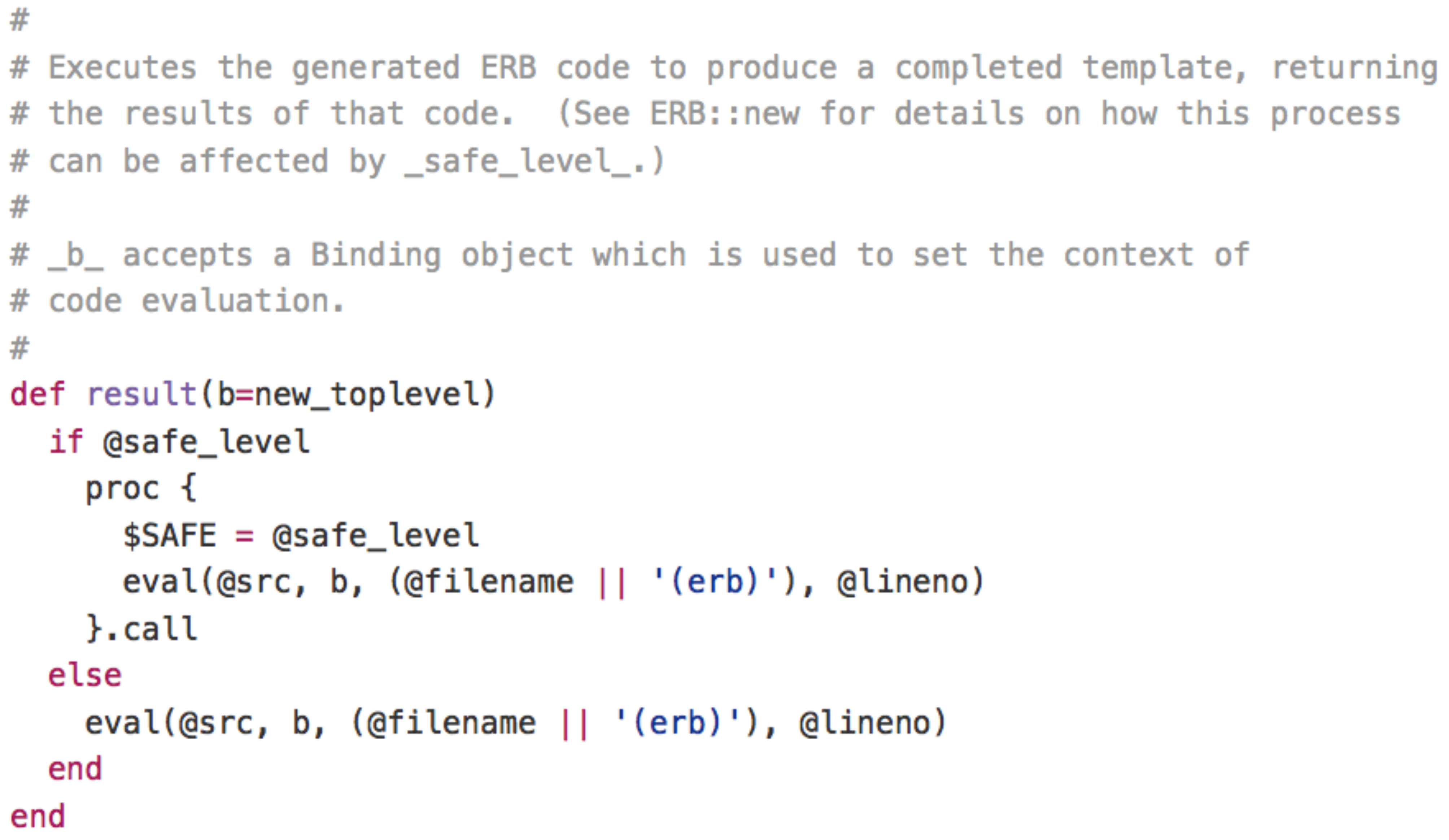
```
def hard_mix(fg, bg, opts={})
  return apply_opacity(fg, opts)
    if fully_transparent?(bg)

  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
```

```ruby
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
```

```
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
```

```ruby
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
```



```
#
# Executes the generated ERB code to produce a completed template, returning
# the results of that code.  (See ERB::new for details on how this process
# can be affected by _safe_level_.)
#
# _b_ accepts a Binding object which is used to set the context of
# code evaluation.
#
def result(b=new_toplevel)
  if @safe_level
    proc {
      $SAFE = @safe_level
      eval(@src, b, (@filename || '(erb)'), @lineno)
    }.call
  else
    eval(@src, b, (@filename || '(erb)'), @lineno)
  end
end
```

```
require 'benchmark/ips'

Benchmark.ips do |x|
  x.report( "direct",      "14 + 2"          )
  x.report( "send-symbol", "14.send(:+, 2)"  )
  x.report( "send-string", "14.send('+', 2)" )
  x.report( "eval",        "eval('14 + 2')"  )
  x.compare!
end
```

```
     direct: 37299872.6 i/s
send-symbol: 13060179.1 i/s - 2.86x slower
send-string:  4974575.3 i/s - 7.50x slower
       eval:   171835.9 i/s - 217.07x slower
```

Throwing away metaprogramming

- Crystal – throws away metaprogramming entirely to make a faster Ruby
- Rubinius – doesn’t support set_trace_func
- JRuby – doesn’t support set_trace_func or ObjectSpace
- RubyMotion – doesn’t support eval, Binding
- A real shame, and not necessary

How does Truffle + Graal solve this?

Escape analysis and partial evaluation

```
def min(a, b)
  [a, b].sort[0]
end


puts min(2, 8)
```

```
~~def min(a, b)~~
~~  [a, b].sort[0]~~
~~end~~


puts [2, 8].sort[0]
```

```
t0 = 2 <=> 8
t1 = t0 < 0 ? 2 : 8
t2 = t0 > 0 ? 8 : 2
t3 = [t1, t2]

puts t3[0]
```

```
t0 = 2 <=> 8
t1 = t0 < 0 ? 2 : 8
~~t2 = t0 > 0 ? 8 : 2~~
~~t3 = [t1, t2]~~

puts t1
```

```
t0 = -1
t1 = t0 < 0 ? 2 : 8

puts t1
```

```
~~t0 = -1~~
t1 = -1 < 0 ? 2 : 8


puts t1
```

```
t1 = true ? 2 : 8

puts t1
```

```
t1 = 2

puts t1
```

```
~~t1 = 2~~

puts 2
puts 2
```

```
t0 = a <=> b
t1 = t0 < 0 ? a : b


puts t1
```

~~t0 = a <=> b~~

```
t1 = (a <=> b) < 0 ? a : b


puts t1
```

~~`t1 = (a <=> b) < 0 ? a : b`~~

```
puts (a <=> b) < 0 ? a : b
puts (a <=> b) < 0 ? a : b
```

```
require 'benchmark/ips'

Benchmark.ips do |x|
  x.report( "direct",      "14 + 2"          )
  x.report( "send-symbol", "14.send(:+, 2)"  )
  x.report( "send-string", "14.send('+', 2)" )
  x.report( "eval",        "eval('14 + 2')"  )
  x.compare!
end
```

```
     direct: 37299872.6 i/s
send-symbol: 13060179.1 i/s - 2.86x slower
send-string:  4974575.3 i/s - 7.50x slower
       eval:   171835.9 i/s - 217.07x slower
     direct: 73099792.5 i/s
send-symbol: 73458837.7 i/s – difference within err
send-string: 66882023.8 i/s - difference within err
       eval: 67024838.3 i/s - difference within err
```

An extreme example

```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
```

```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
```

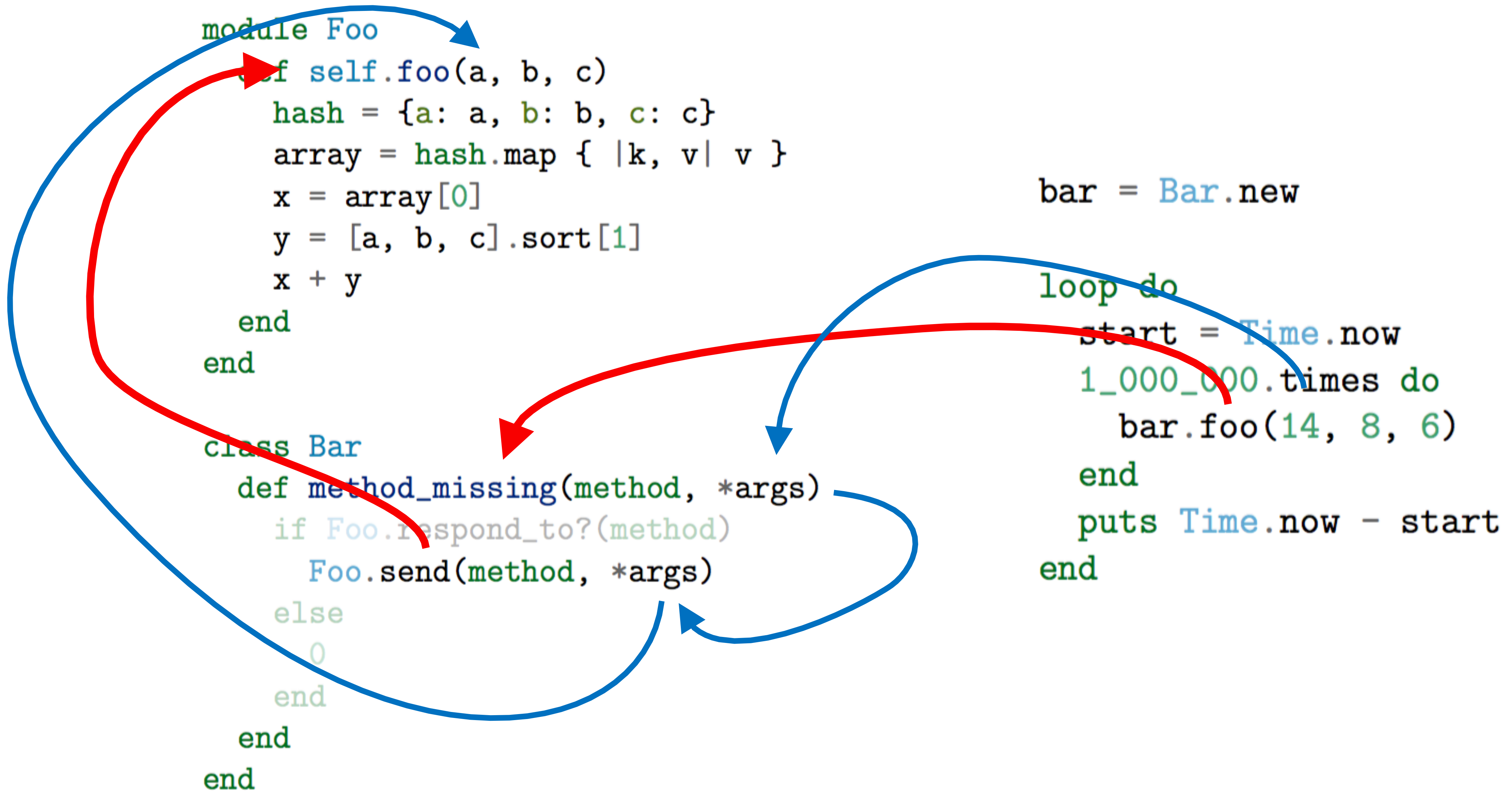
```ruby
bar = Bar.new

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

```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
```

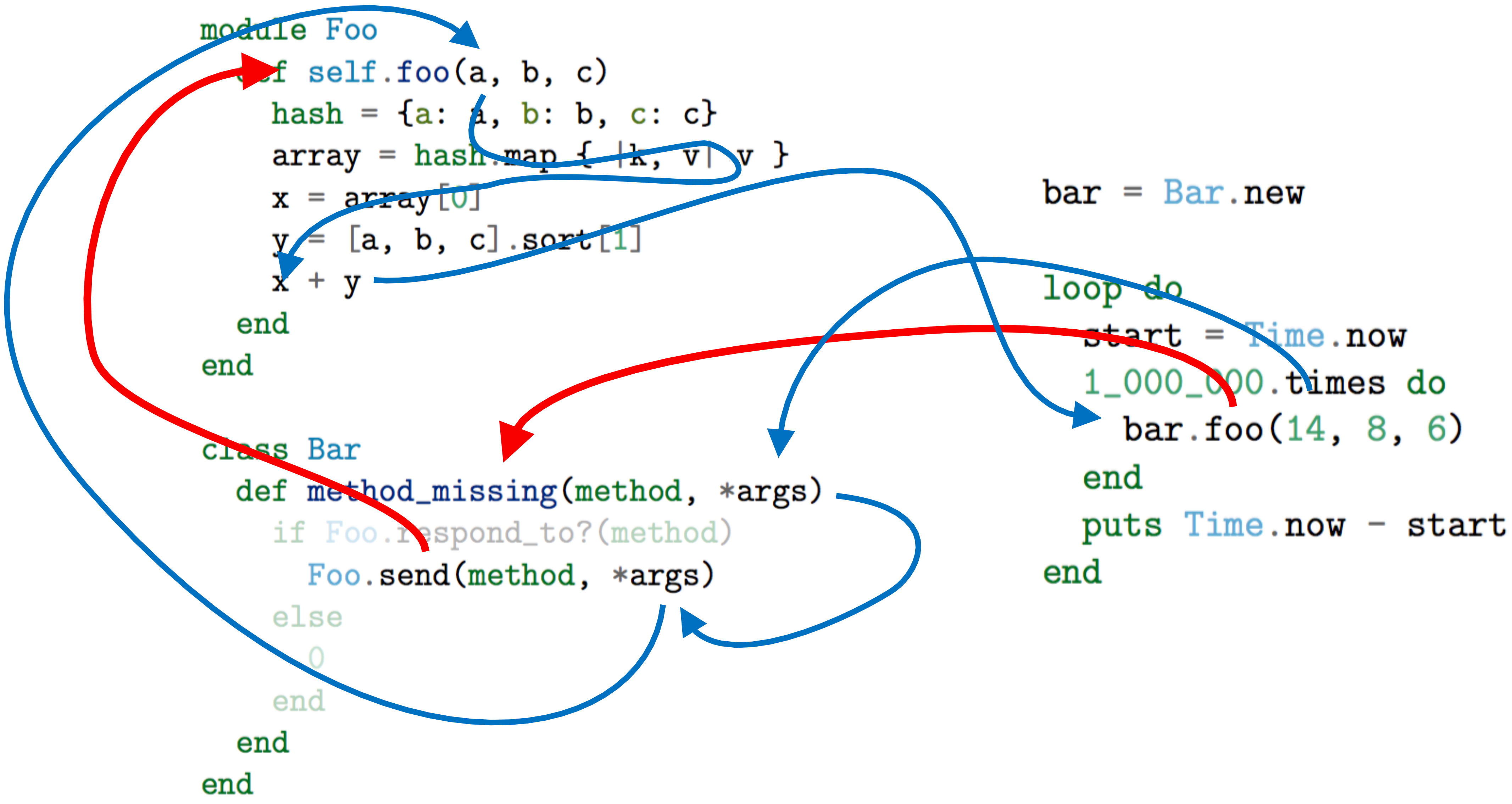
```
module Foo
  def self.foo(a, b, c)
    hash = {a: a, b: b, c: c}
    array = hash.map { |k, v| v }
    x = array[0]
    y = [a, b, c].sort[1]
    x + y
  end
end

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

```
bar = Bar.new

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



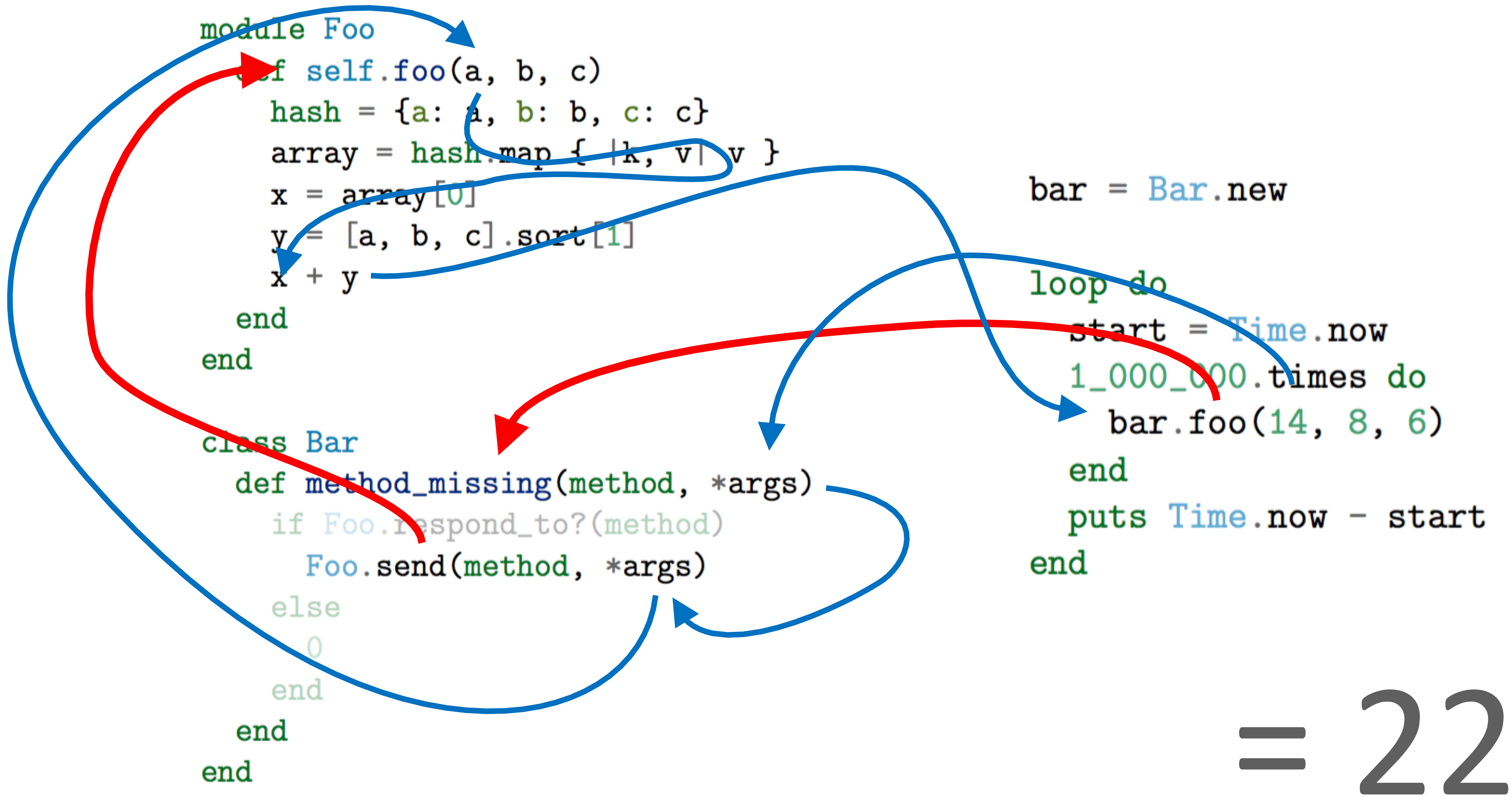
```
module Foo
  def self.foo(a, b, c)
    hash = {a: a, b: b, c: c}
    array = hash.map { |k, v| v }
    x = array[0]
    y = [a, b, c].sort[1]
    x + y
  end
end

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

```
bar = Bar.new

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



```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
```

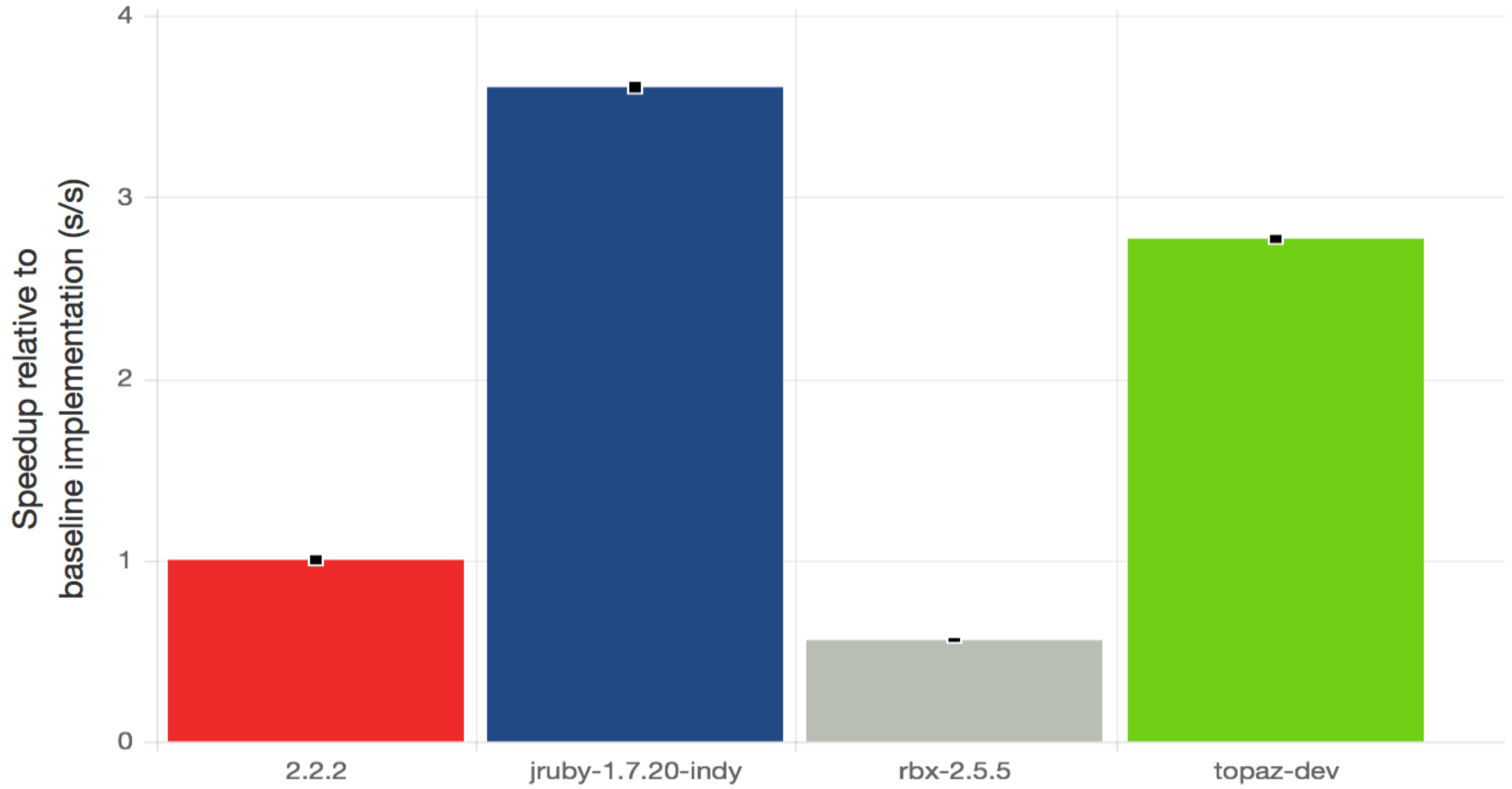
```ruby
bar = Bar.new

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



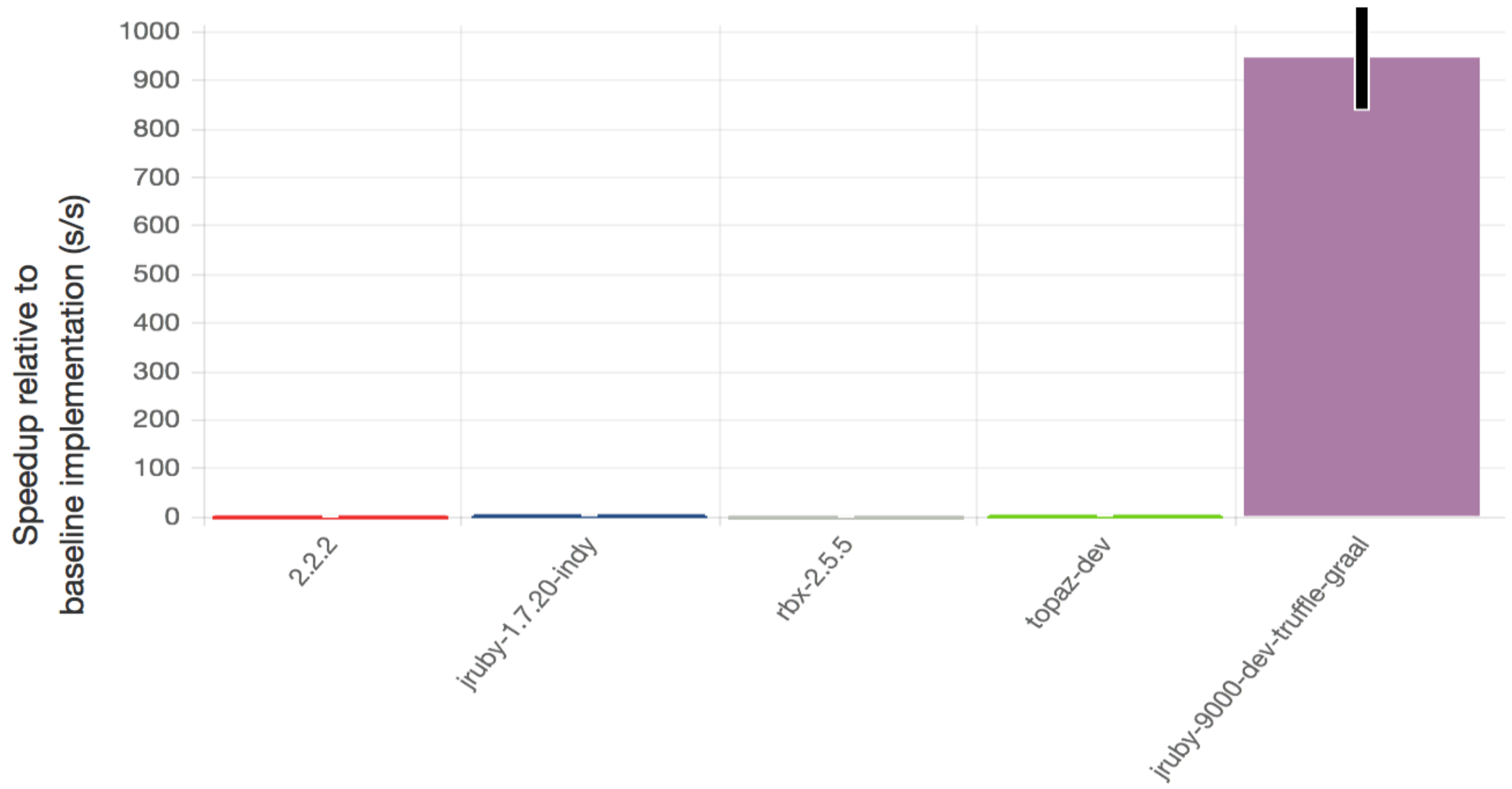
```
module Foo
  def self.foo(a, b, c)
    hash = {a: a, b: b, c: c}
    array = hash.map { |k, v| v }
    x = array[0]
    y = [a, b, c].sort[1]
    x + y
  end
end

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

```
bar = Bar.new

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

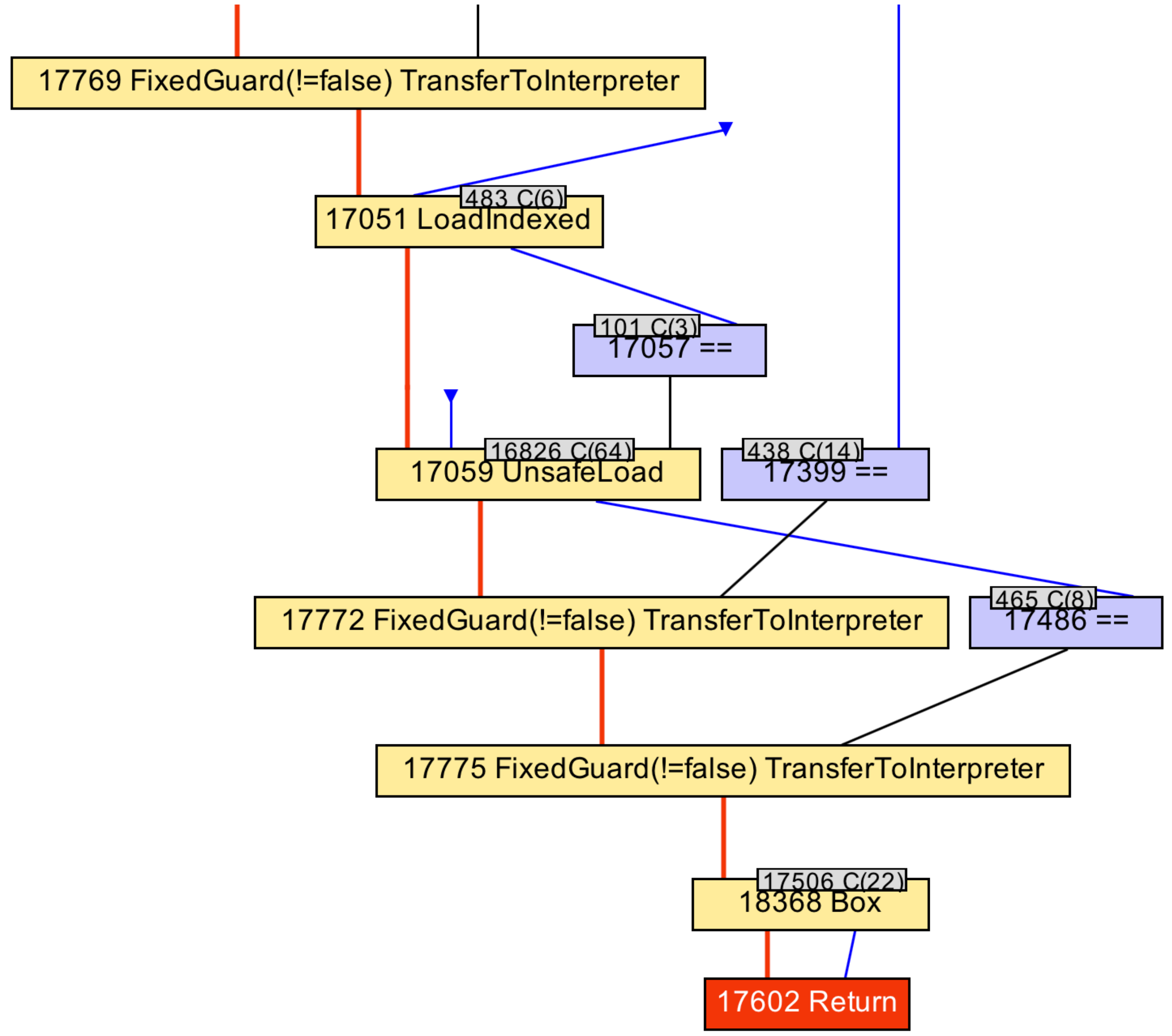
= 22 !

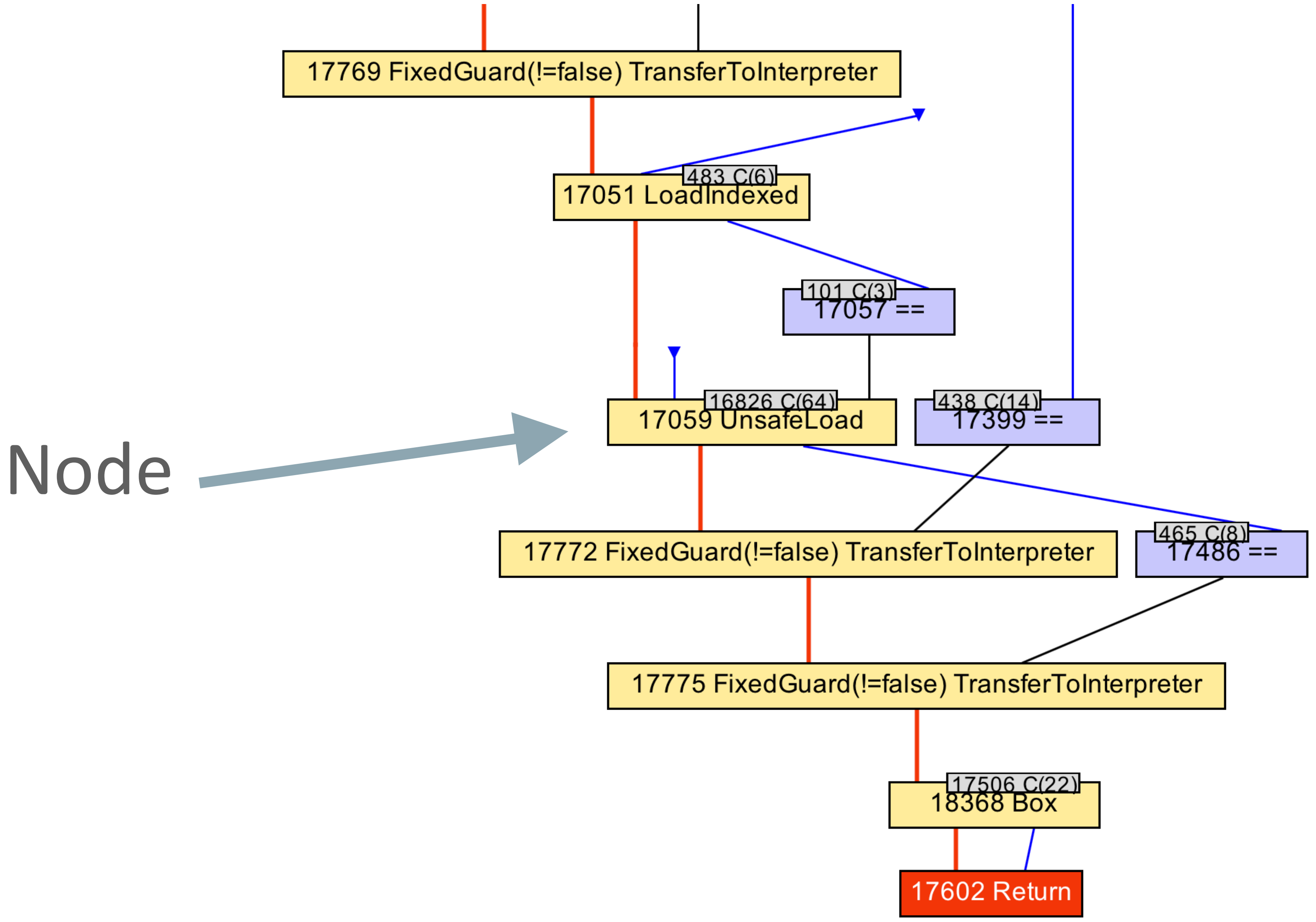


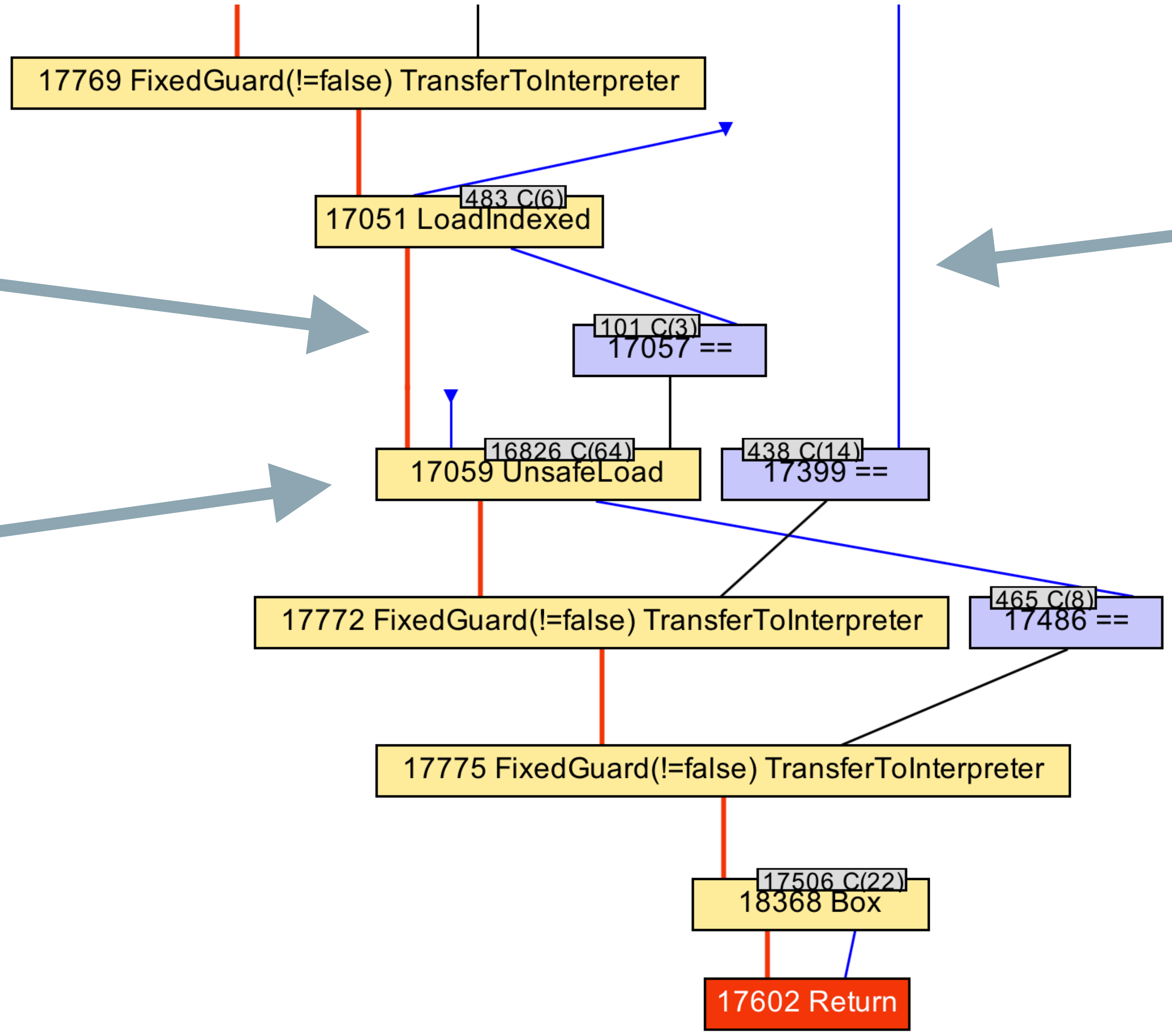
Speedup relative to baseline implementation (s/s)
4
3
2
1
0
2.2.2
jruby-1.7.20-indy
rbx-2.5.5
topaz-dev

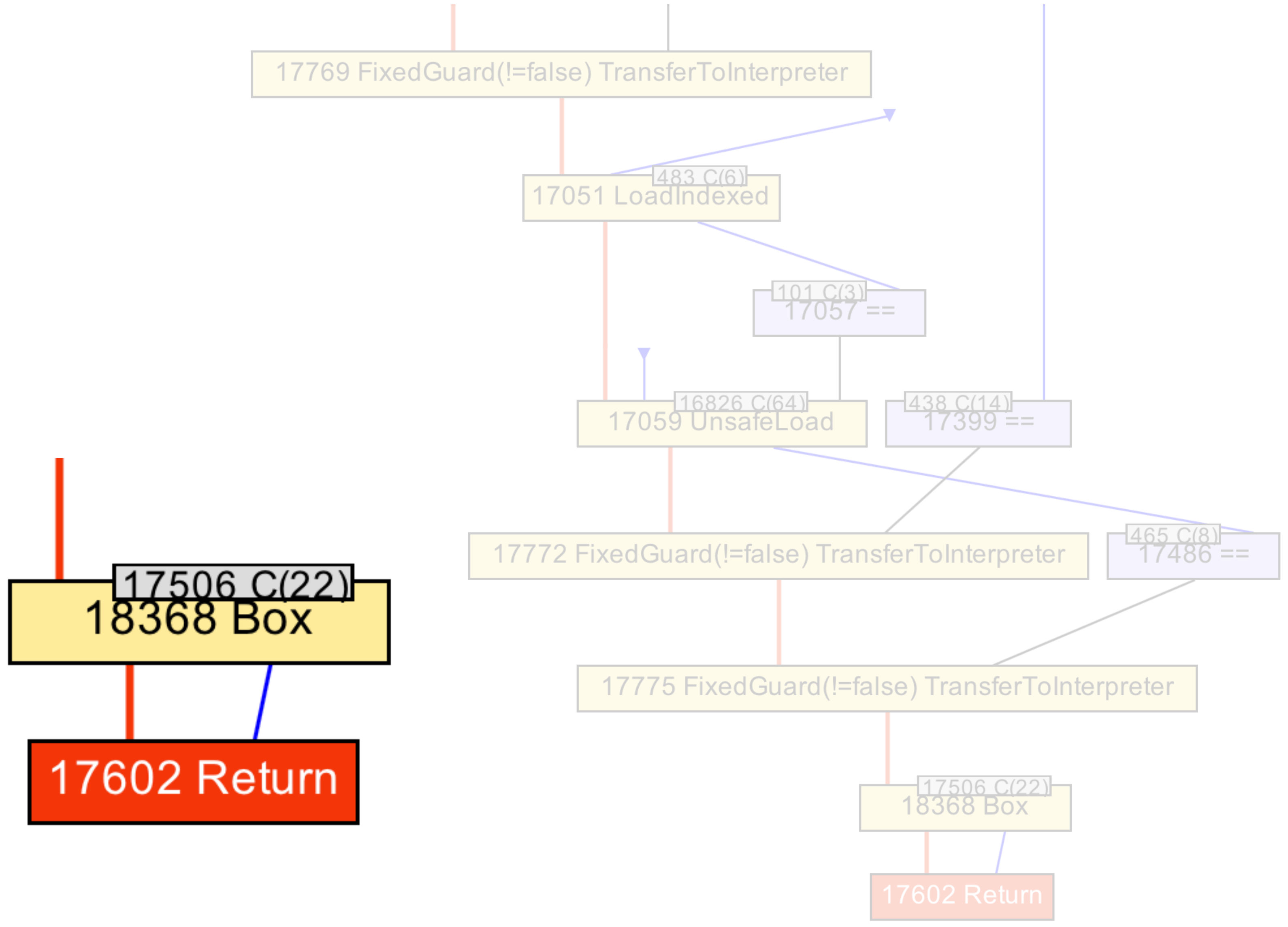


Speedup relative to baseline implementation (s/s)
1000
900
800
700
600
500
400
300
200
100
0
2.2.2
jruby-1.7.20-indy
rbx-2.5.5
topaz-dev
jruby-9000-dev-truffle-graal

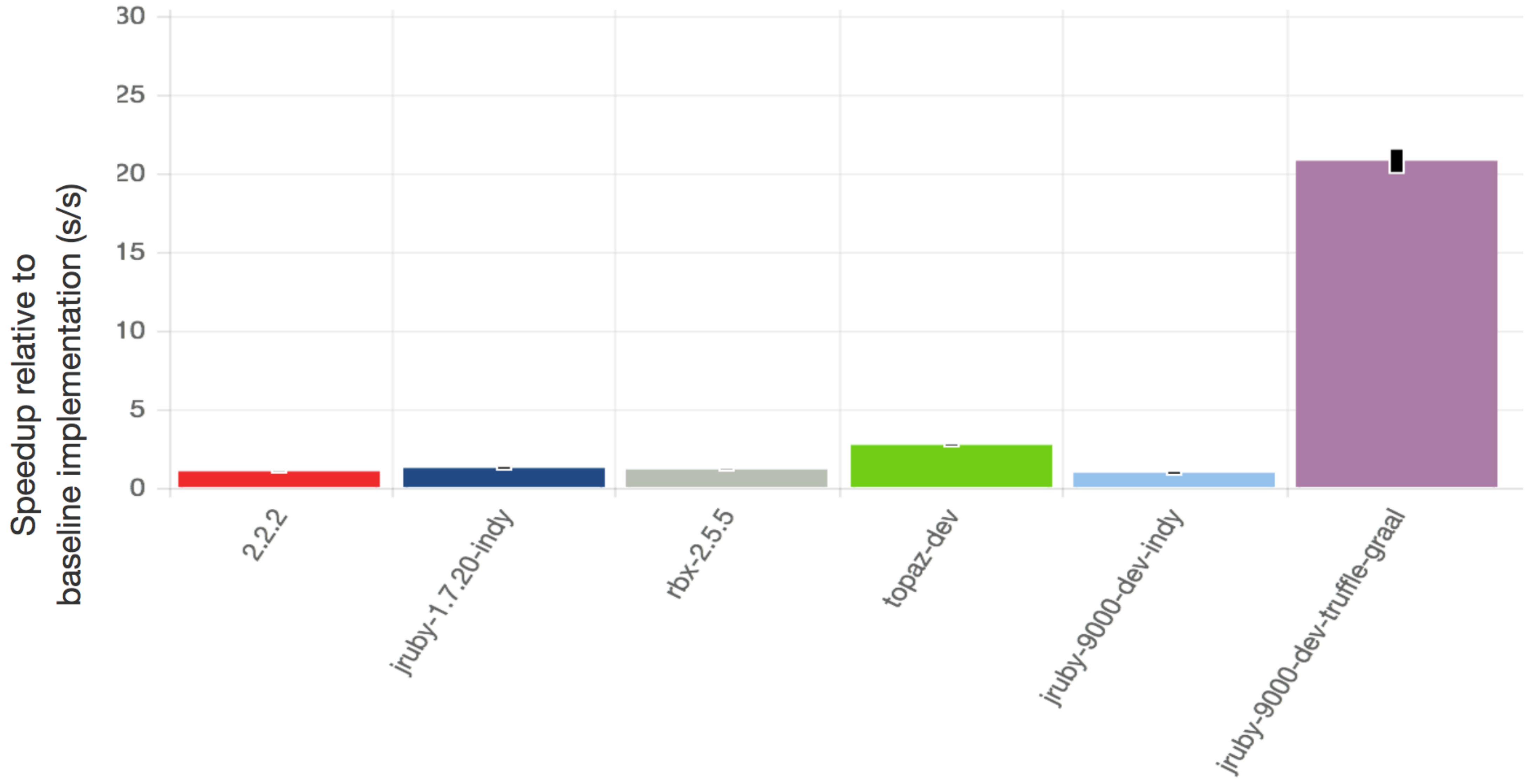

17769 FixedGuard(!=false) TransferToInterpreter
483 C(6)
17051 LoadIndexed
101 C(3)
17057 ==
16826 C(64)
17059 UnsafeLoad
438 C(14)
17399 ==
17772 FixedGuard(!=false) TransferToInterpreter
465 C(8)
17486 ==
17775 FixedGuard(!=false) TransferToInterpreter
17506 C(22)
18368 Box
17602 Return


Node
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


Control flow
Node
Data flow
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
```

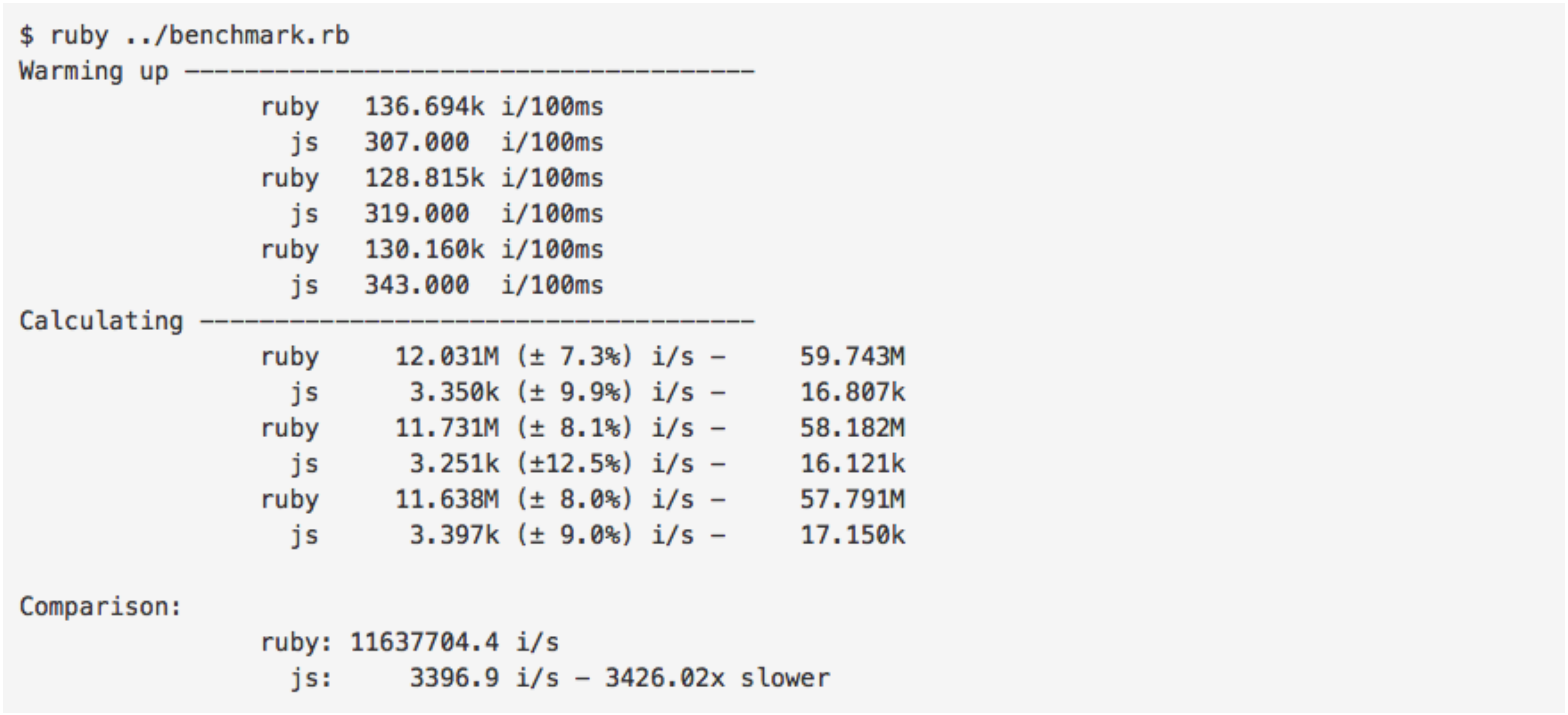

Speedup relative to baseline implementation (s/s)
30
25
20
15
10
5
0
2.2.2
jruby-1.7.20-indy
rbx-2.5.5
topaz-dev
jruby-9000-dev-indy
jruby-9000-dev-truffle-graal

ORACLE

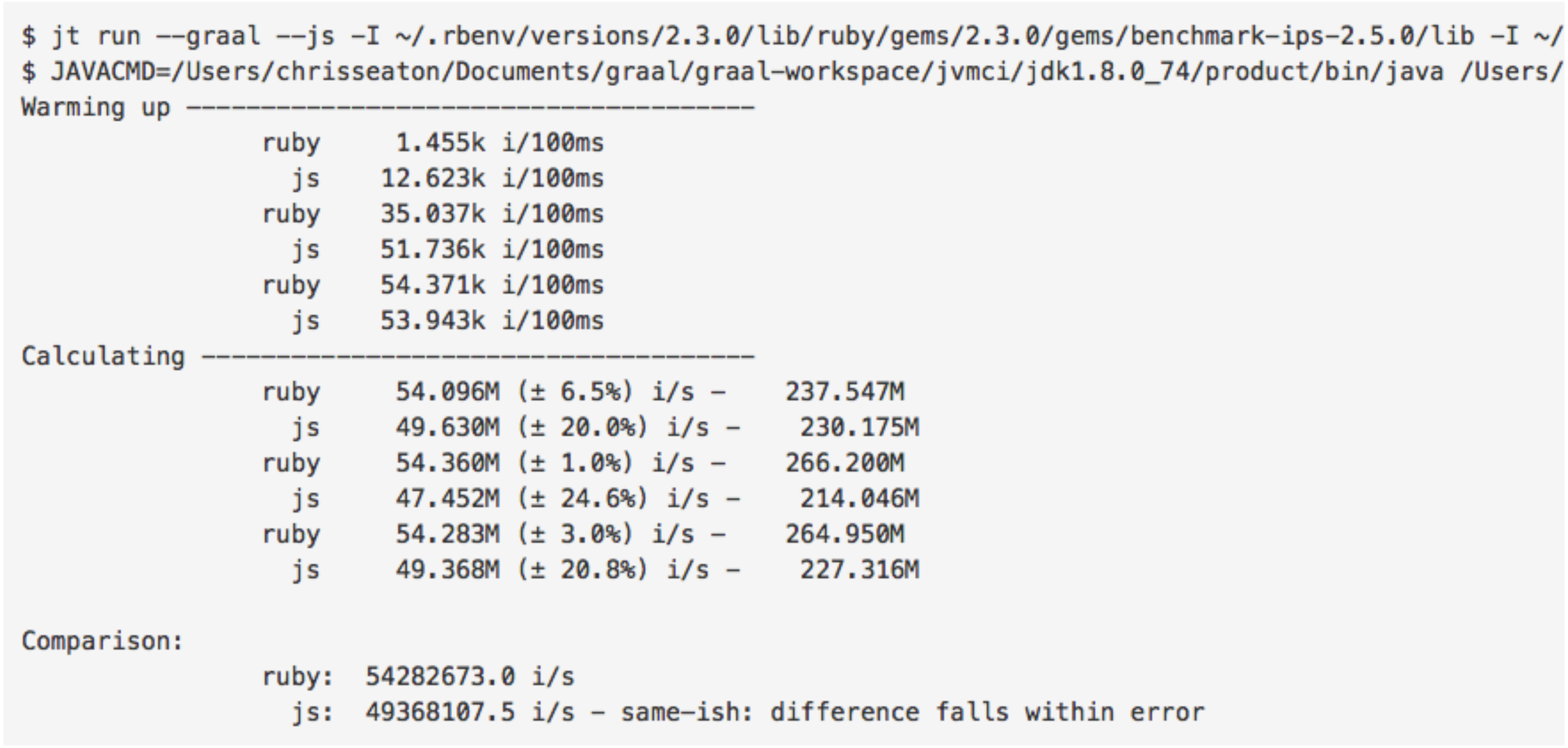
Polyglot

```
14 + 2
```

```
ExecJS.eval('14 + 2')
```



```
$ ruby ../benchmark.rb
Warming up --------------------------------------
                ruby   136.694k i/100ms
                  js   307.000  i/100ms
                ruby   128.815k i/100ms
                  js   319.000  i/100ms
                ruby   130.160k i/100ms
                  js   343.000  i/100ms
Calculating --------------------------------------
                ruby     12.031M (± 7.3%) i/s -     59.743M
                  js      3.350k (± 9.9%) i/s -     16.807k
                ruby     11.731M (± 8.1%) i/s -     58.182M
                  js      3.251k (±12.5%) i/s -     16.121k
                ruby     11.638M (± 8.0%) i/s -     57.791M
                  js      3.397k (± 9.0%) i/s -     17.150k

Comparison:
                ruby: 11637704.4 i/s
                  js:     3396.9 i/s - 3426.02x slower
```



```
$ jt run --graal --js -I ~/.rbenv/versions/2.3.0/lib/ruby/gems/2.3.0/gems/benchmark-ips-2.5.0/lib -I ~/
$ JAVACMD=/Users/chrisseaton/Documents/graal/graal-workspace/jvmci/jdk1.8.0_74/product/bin/java /Users/
Warming up --------------------------------------
                ruby     1.455k i/100ms
                  js    12.623k i/100ms
                ruby    35.037k i/100ms
                  js    51.736k i/100ms
                ruby    54.371k i/100ms
                  js    53.943k i/100ms
Calculating -------------------------------------
                ruby     54.096M (± 6.5%) i/s -    237.547M
                  js     49.630M (± 20.0%) i/s -    230.175M
                ruby     54.360M (± 1.0%) i/s -    266.200M
                  js     47.452M (± 24.6%) i/s -    214.046M
                ruby     54.283M (± 3.0%) i/s -    264.950M
                  js     49.368M (± 20.8%) i/s -    227.316M

Comparison:
                ruby:  54282673.0 i/s
                  js:  49368107.5 i/s - same-ish: difference falls within error
```

Conclusions

We don’t need to tell Ruby, JS, R etc programmers to avoid language features to get performance

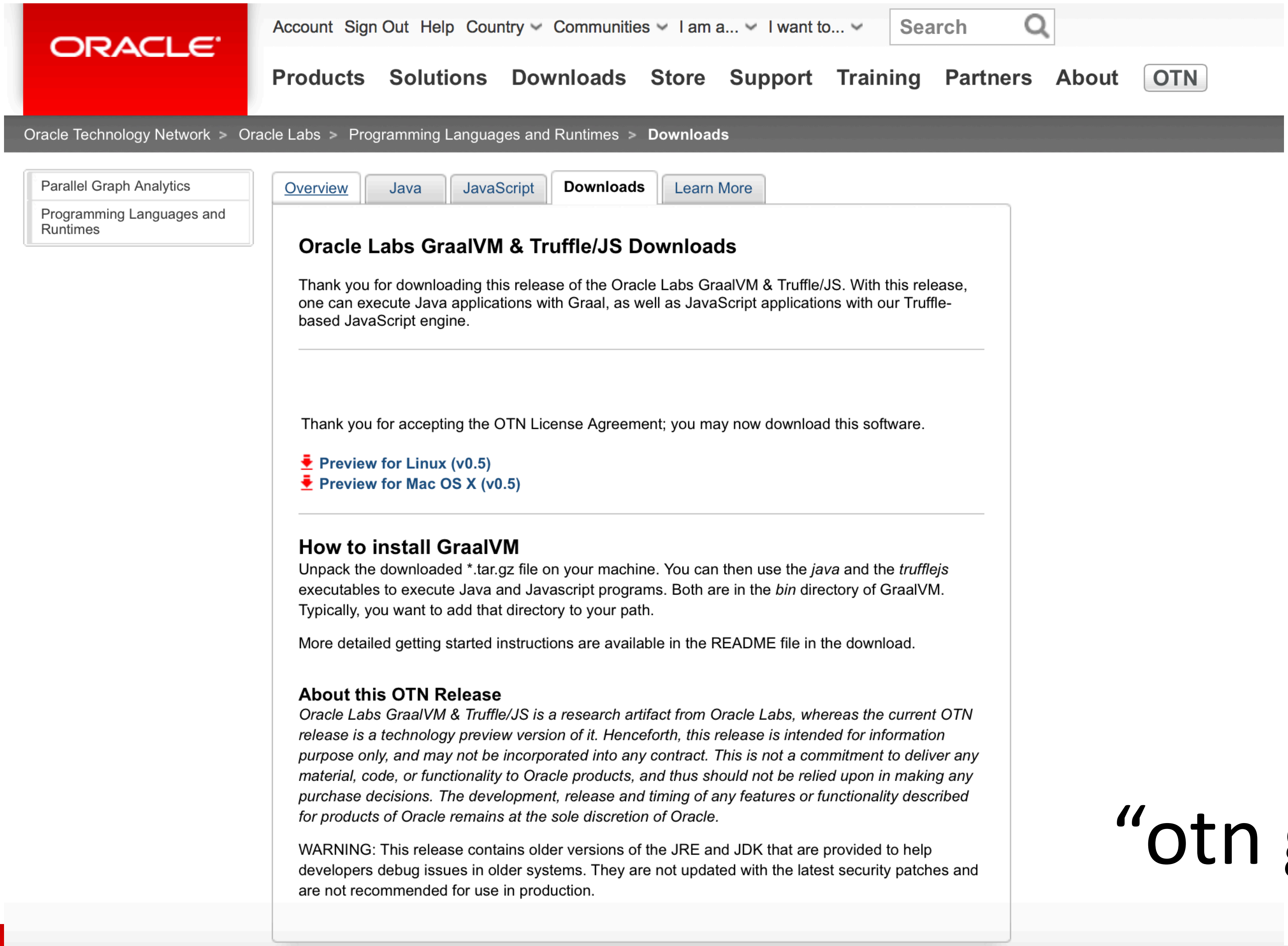
Conclusions

We don’t need to tell Ruby, JS, R etc programmers to avoid language features to get performance

We can make a better JIT compiler by writing it in Java

Where to get more info

ORACLE®


Account Sign Out Help Country Communities I am a... I want to...
Search
Products Solutions Downloads Store Support Training Partners About OTN
Oracle Technology Network > Oracle Labs > Programming Languages and Runtimes > Downloads
Parallel Graph Analytics
Programming Languages and Runtimes
Overview Java JavaScript Downloads Learn More

Oracle Labs GraalVM & Truffle/JS Downloads

Thank you for downloading this release of the Oracle Labs GraalVM & Truffle/JS. With this release, one can execute Java applications with Graal, as well as JavaScript applications with our Truffle-based JavaScript engine.

Thank you for accepting the OTN License Agreement; you may now download this software.

Preview for Linux (v0.5)
Preview for Mac OS X (v0.5)

How to install GraalVM

Unpack the downloaded *.tar.gz file on your machine. You can then use the *java* and the *trufflejs* executables to execute Java and Javascript programs. Both are in the *bin* directory of GraalVM. Typically, you want to add that directory to your path.

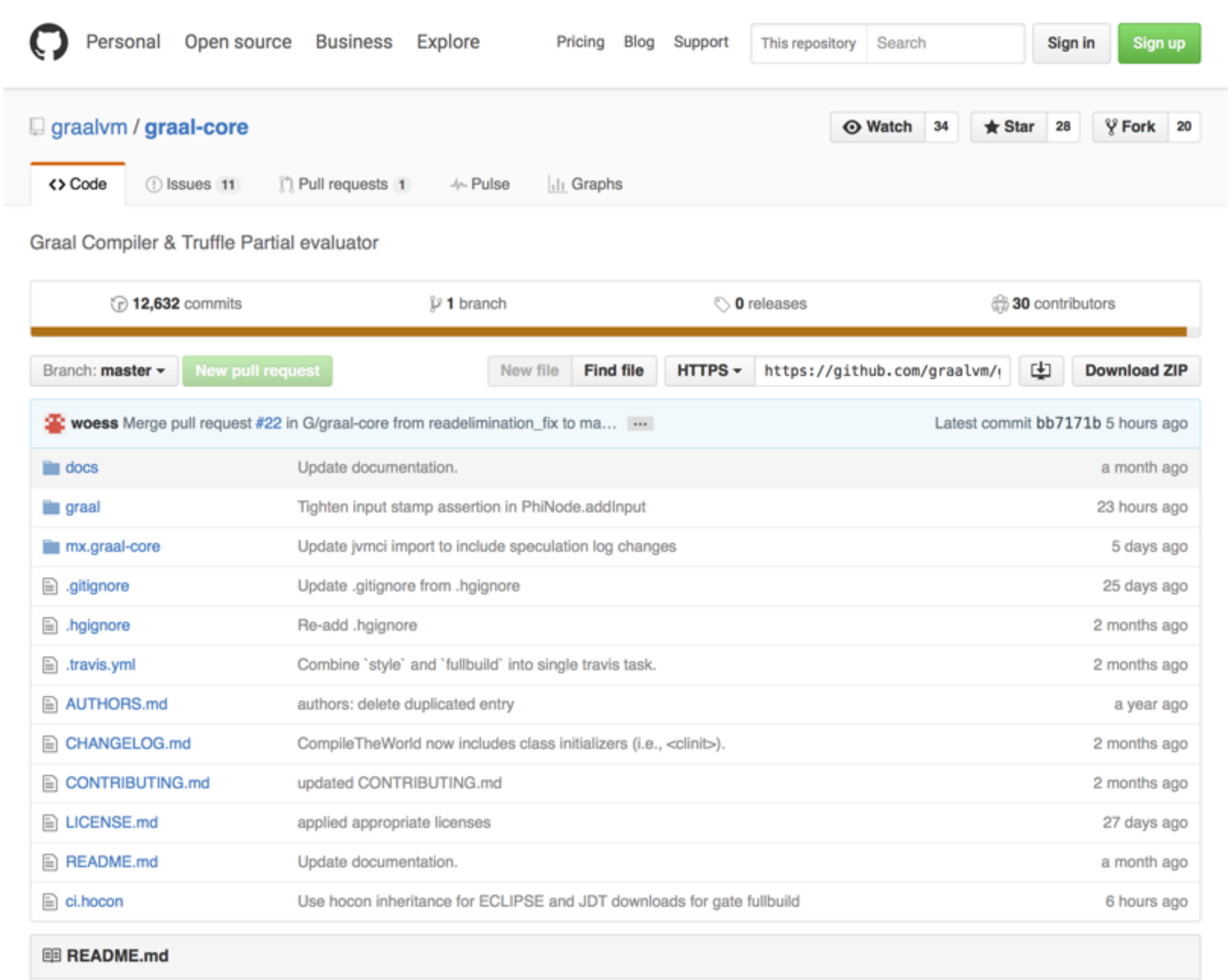
More detailed getting started instructions are available in the README file in the download.

About this OTN Release

Oracle Labs GraalVM & Truffle/JS is a research artifact from Oracle Labs, whereas the current OTN release is a technology preview version of it. Henceforth, this release is intended for information purpose only, and may not be incorporated into any contract. This is not a commitment to deliver any material, code, or functionality to Oracle products, and thus should not be relied upon in making any purchase decisions. The development, release and timing of any features or functionality described for products of Oracle remains at the sole discretion of Oracle.

WARNING: This release contains older versions of the JRE and JDK that are provided to help developers debug issues in older systems. They are not updated with the latest security patches and are not recommended for use in production.

"otn graal"



Graal Compiler & Truffle Partial evaluator

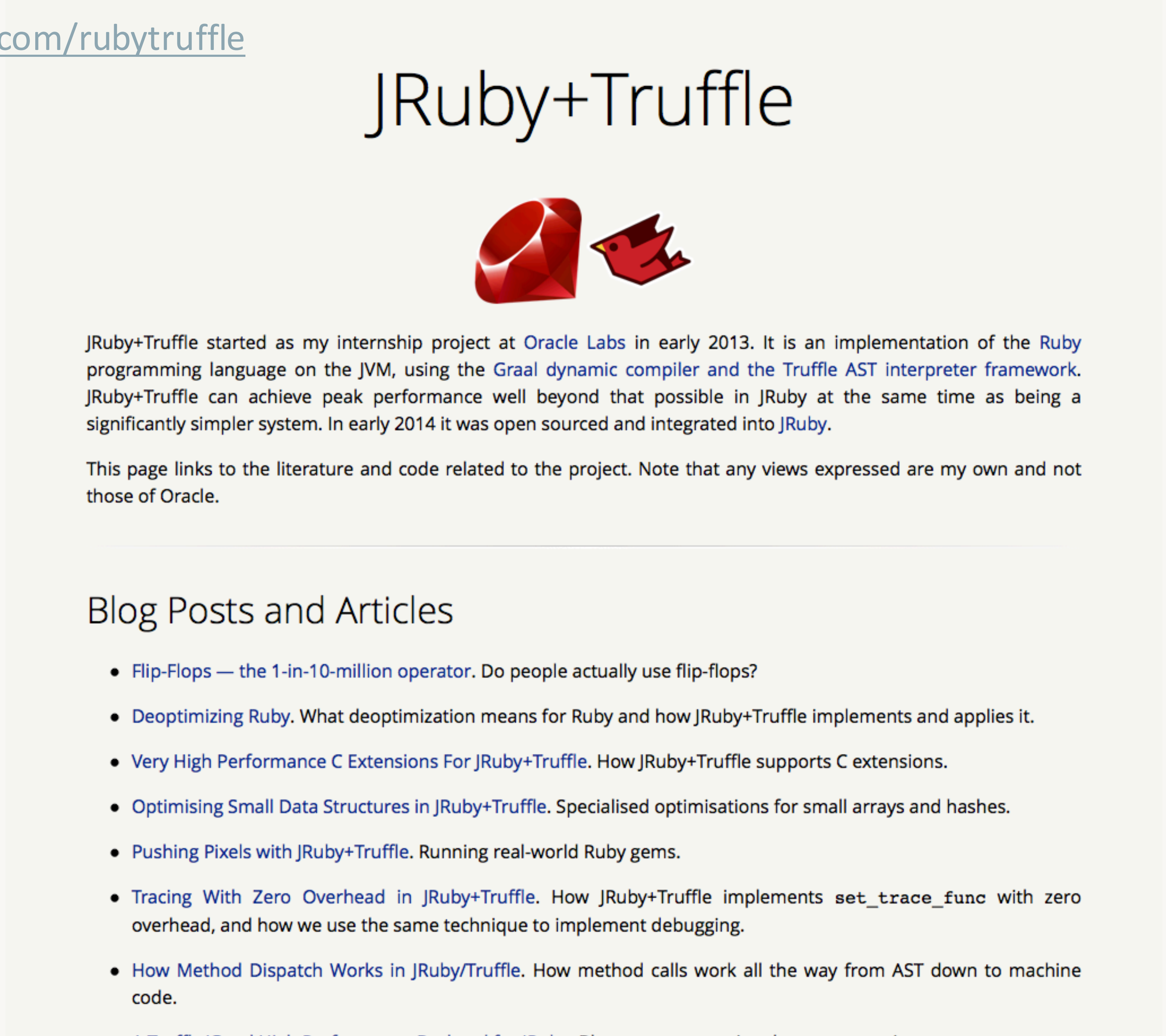
12,632 commits	1 branch	0 releases	30 contributors

woess Merge pull request #22 in G/graal-core from readelimination_fix to ma… Latest commit bb7171b 5 hours ago

docs	Update documentation.	a month ago
graal	Tighten input stamp assertion in PhiNode.addInput	23 hours ago
mx.graal-core	Update jvmci import to include speculation log changes	5 days ago
.gitignore	Update .gitignore from .hgignore	25 days ago
.hgignore	Re-add .hgignore	2 months ago
.travis.yml	Combine `style` and `fullbuild` into single travis task.	2 months ago
AUTHORS.md	authors: delete duplicated entry	a year ago
CHANGELOG.md	CompileTheWorld now includes class initializers (i.e., <clinit>).	2 months ago
CONTRIBUTING.md	updated CONTRIBUTING.md	2 months ago
LICENSE.md	applied appropriate licenses	27 days ago
README.md	Update documentation.	a month ago
ci.hocon	Use hocon inheritance for ECLIPSE and JDT downloads for gate fullbuild	6 hours ago

README.md

http://chrisseaton.com/rubytruffle



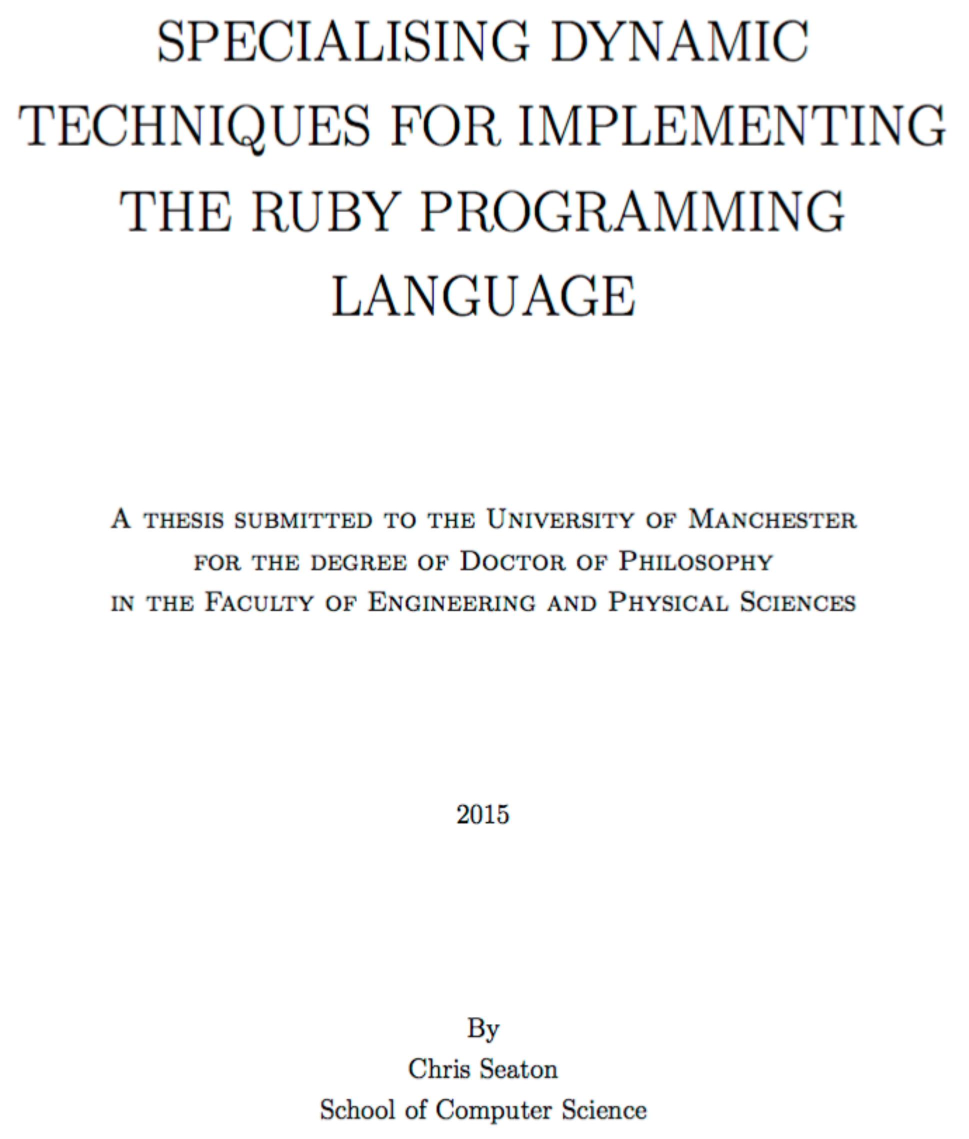
JRuby+Truffle

JRuby+Truffle started as my internship project at Oracle Labs in early 2013. It is an implementation of the Ruby programming language on the JVM, using the Graal dynamic compiler and the Truffle AST interpreter framework. JRuby+Truffle can achieve peak performance well beyond that possible in JRuby at the same time as being a significantly simpler system. In early 2014 it was open sourced and integrated into JRuby.

This page links to the literature and code related to the project. Note that any views expressed are my own and not those of Oracle.

Blog Posts and Articles

- Flip-Flops — the 1-in-10-million operator. Do people actually use flip-flops?
- Deoptimizing Ruby. What deoptimization means for Ruby and how JRuby+Truffle implements and applies it.
- Very High Performance C Extensions For JRuby+Truffle. How JRuby+Truffle supports C extensions.
- Optimising Small Data Structures in JRuby+Truffle. Specialised optimisations for small arrays and hashes.
- Pushing Pixels with JRuby+Truffle. Running real-world Ruby gems.
- Tracing With Zero Overhead in JRuby+Truffle. How JRuby+Truffle implements `set_trace_func` with zero overhead, and how we use the same technique to implement debugging.
- How Method Dispatch Works in JRuby/Truffle. How method calls work all the way from AST down to machine code.



SPECIALISING DYNAMIC TECHNIQUES FOR IMPLEMENTING THE RUBY PROGRAMMING LANGUAGE

A thesis submitted to the University of Manchester
for the degree of Doctor of Philosophy
in the Faculty of Engineering and Physical Sciences

2015

By
Chris Seaton
School of Computer Science

http://chrisseaton.com/phd

Acknowledgements

Benoit Daloze
Kevin Menard
Petr Chalupa

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
Tom Rodriguez
Roland Schatz
Chris Seaton
Doug Simon
Lukas Stadler

Oracle Labs (continued)
Michael Van de Vanter
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
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

chris.seaton@oracle.com

@ChrisGSeaton

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

(or just search for ‘jruby truffle’)

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