
ORACLE®

Graal: A Polyglot VM for a Polyglot IDE

Chris Seaton
Research Manager
Oracle Labs
April 2017

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.

Programming languages

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?

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

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

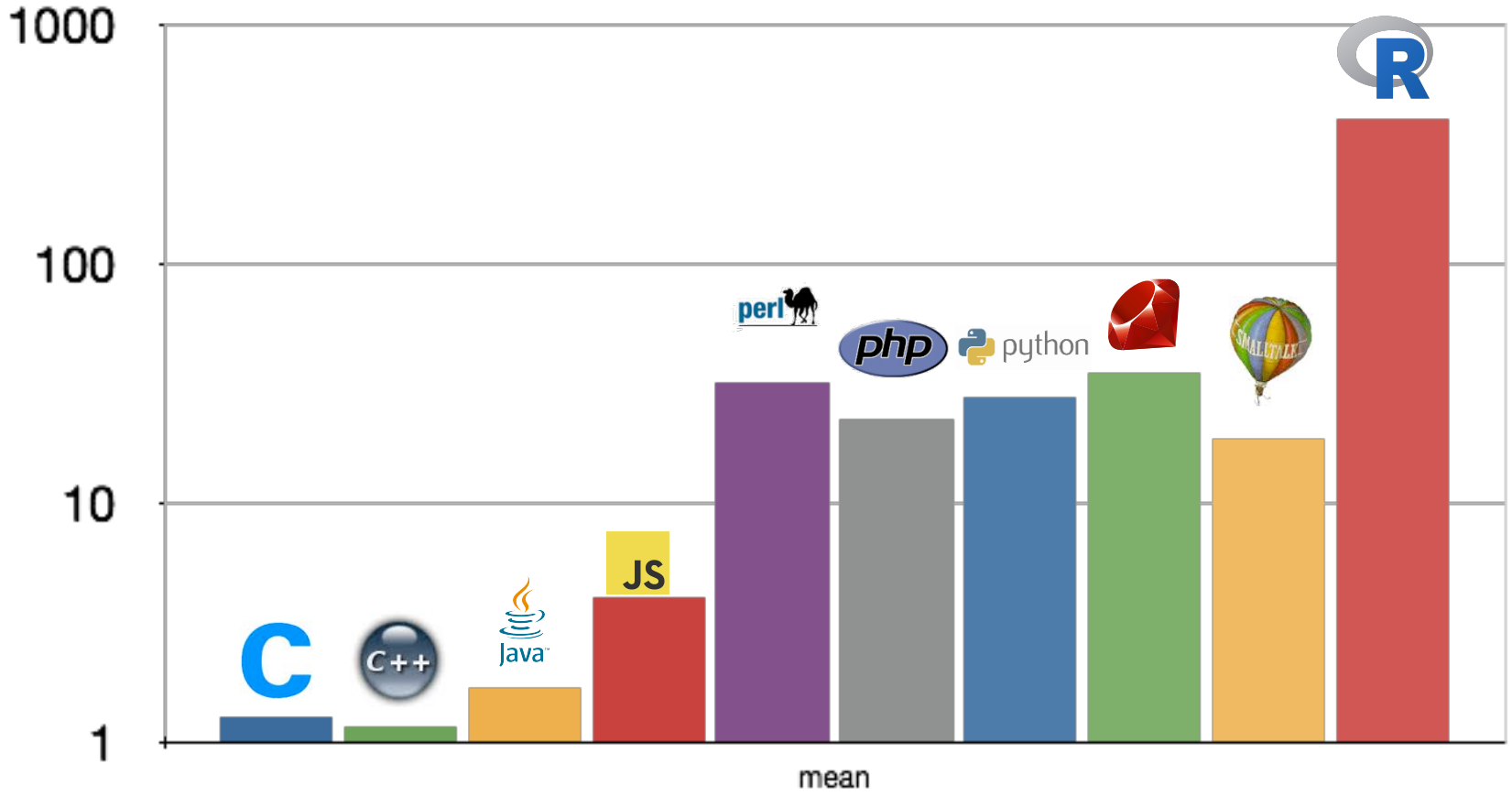
Oct 10, 2005 - **One Language to Rule Them All - Java**. For a long time I'd been hoping to add a scripting language to LibQ, to use in any of my (or other **...**

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

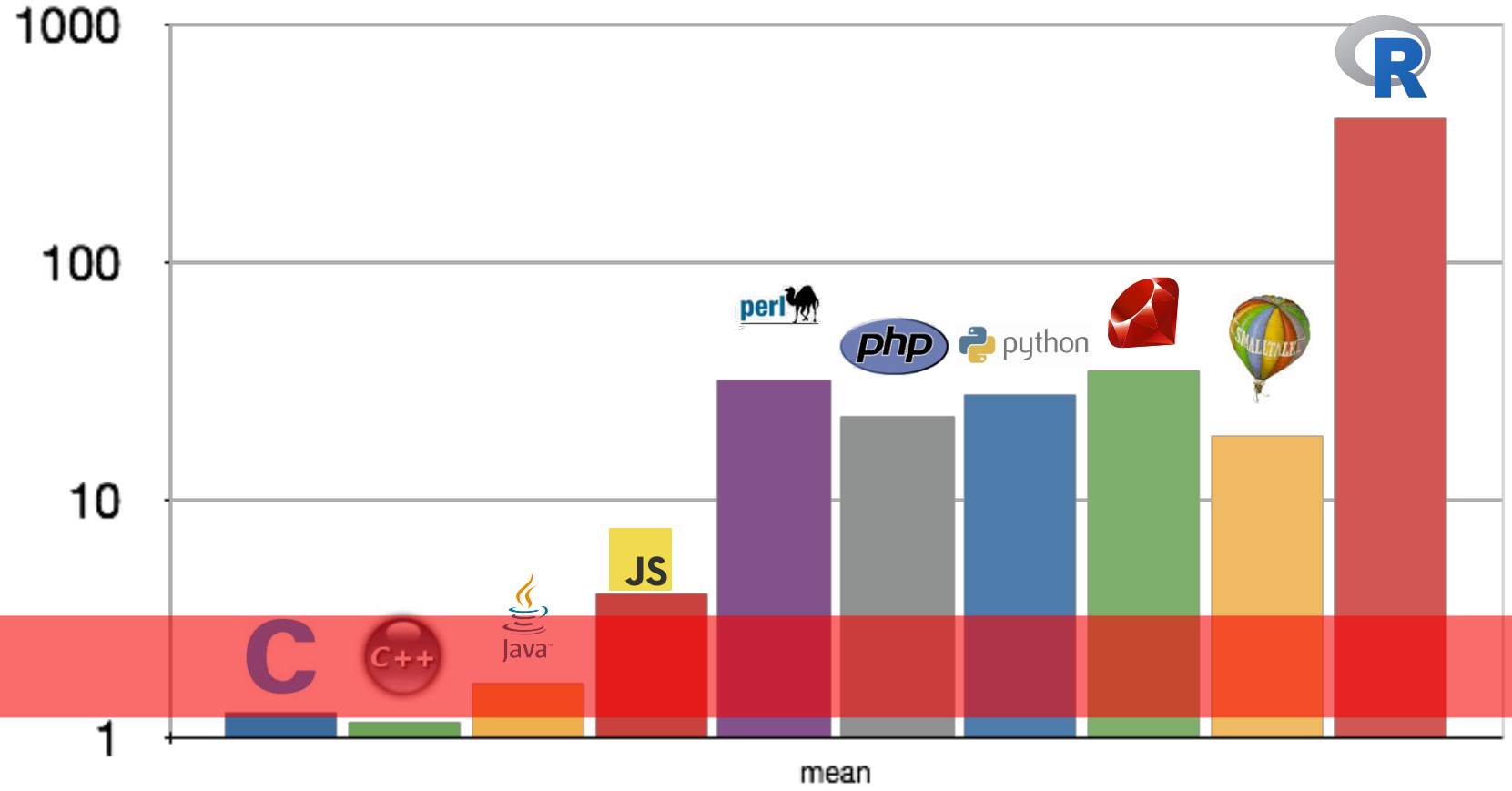
fr.slideshare.net/sdeleuze/dart-mixit2013en

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

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

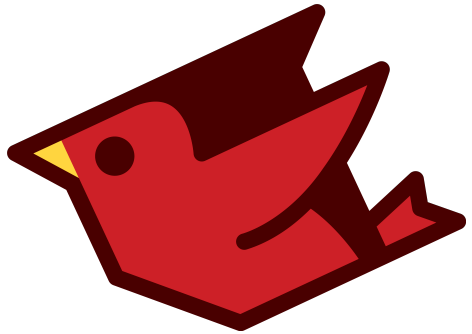
Current situation

- 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

The Graal Approach



Jython

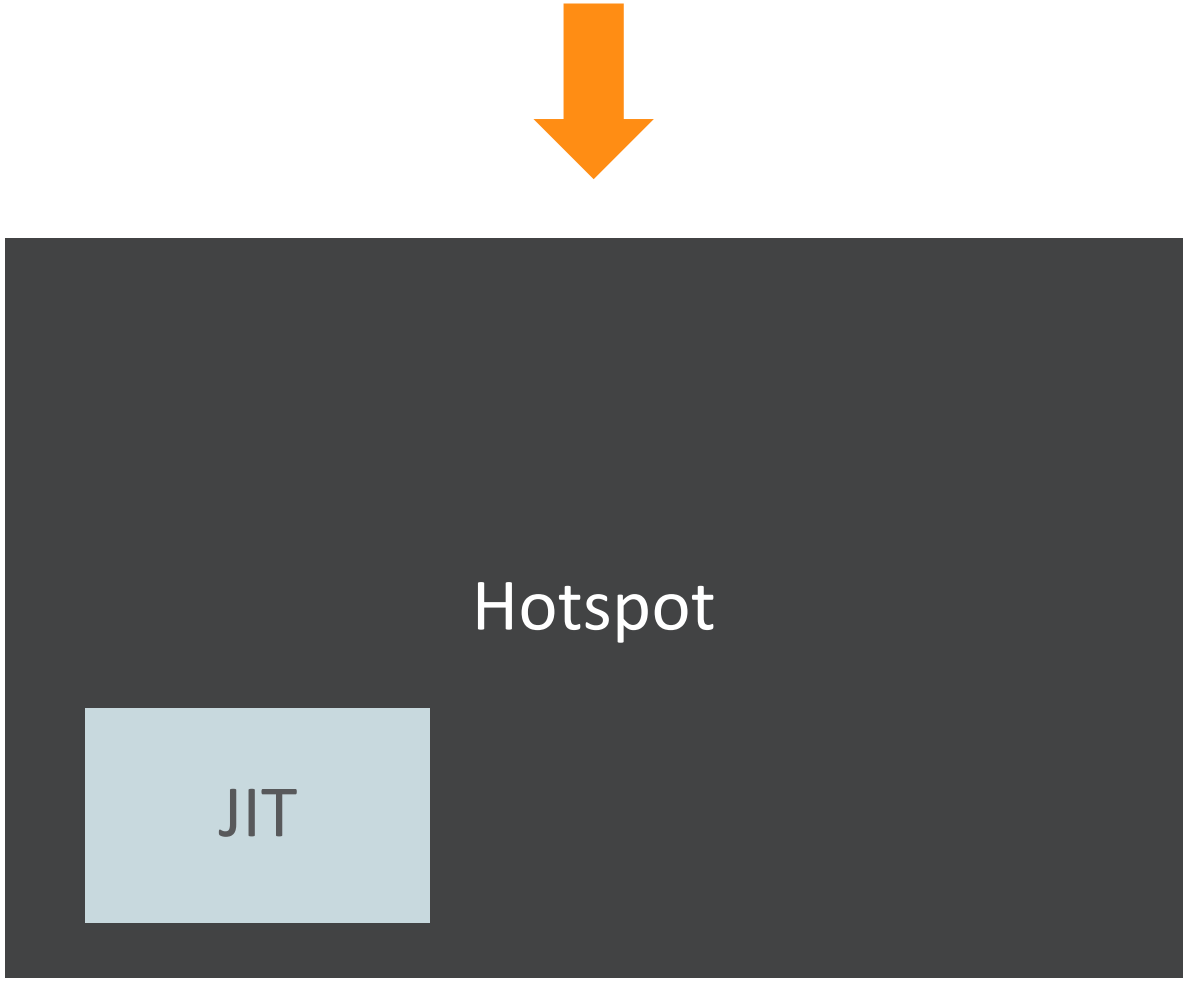
renjin

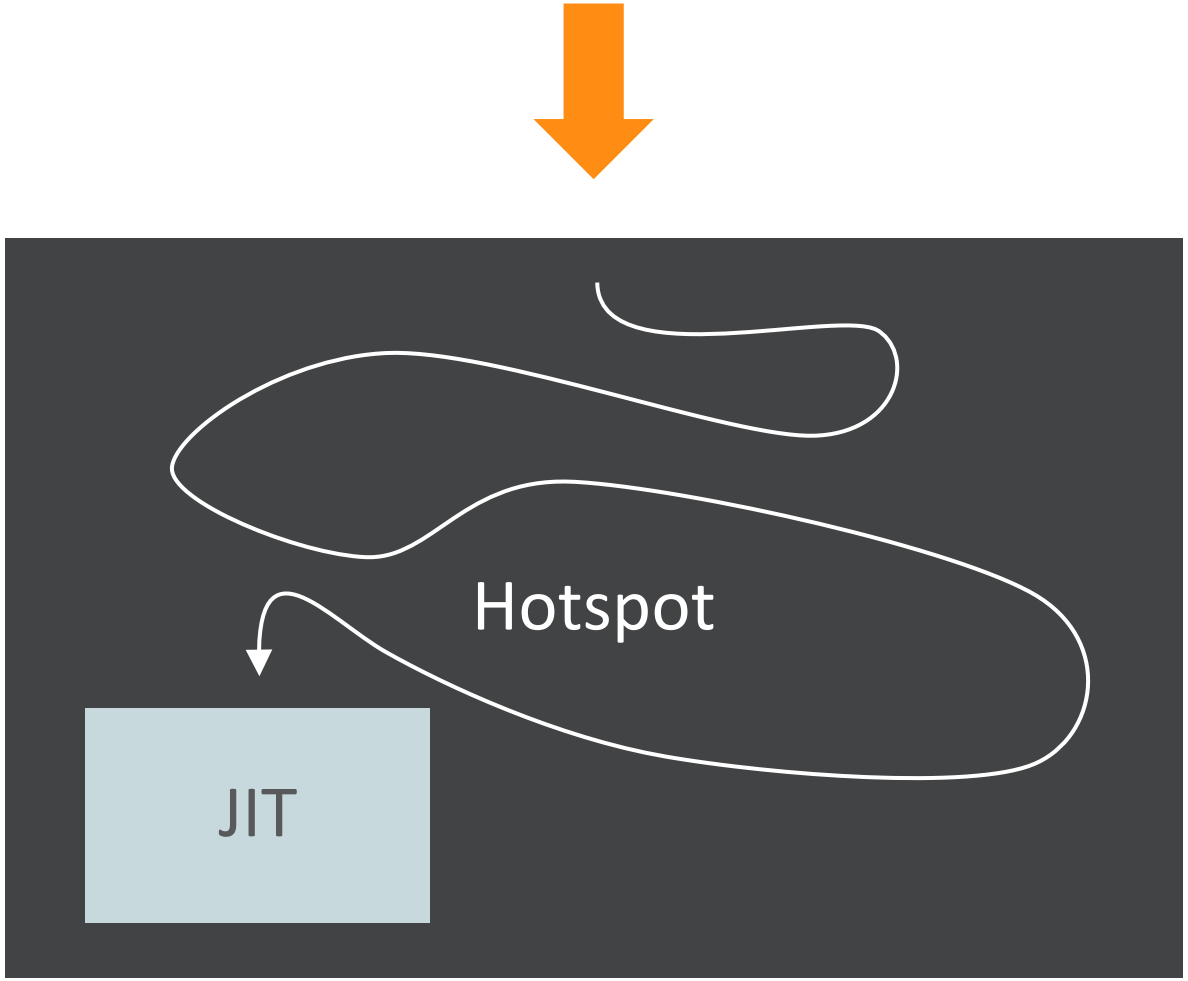
Conventional JVM implementations of languages work by emitting JVM bytecode – the same thing that the Java compiler does

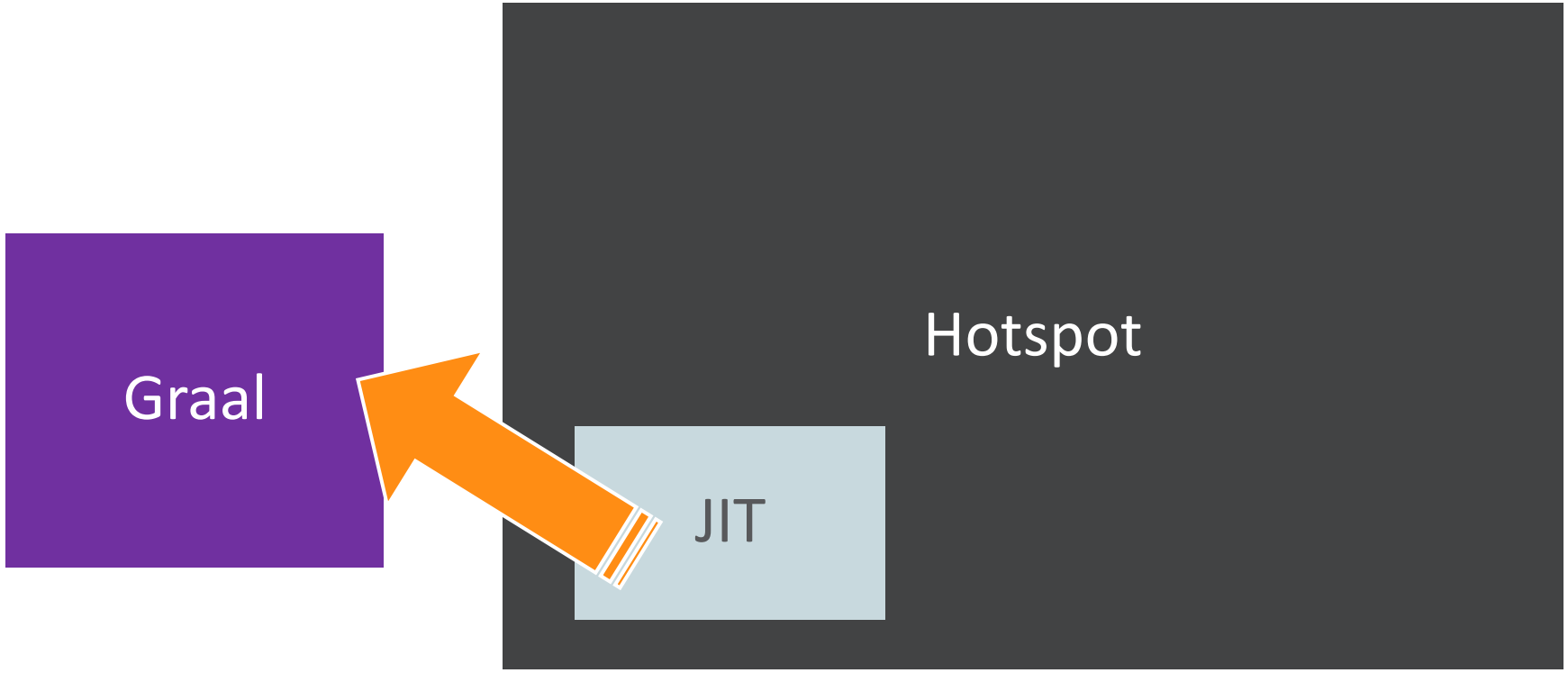
ORACLE

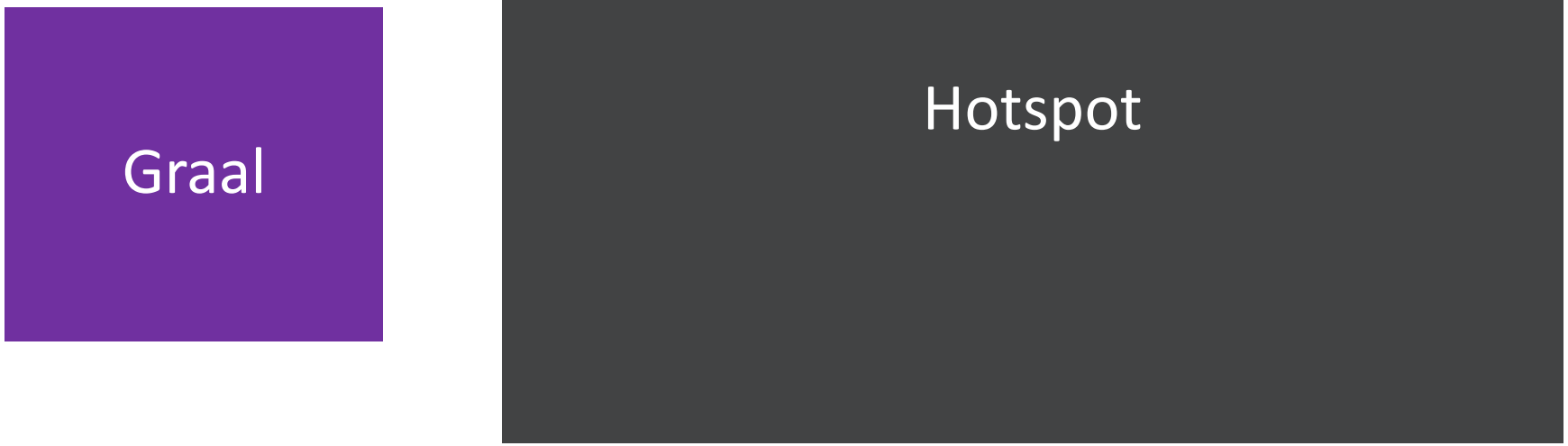
Hotspot

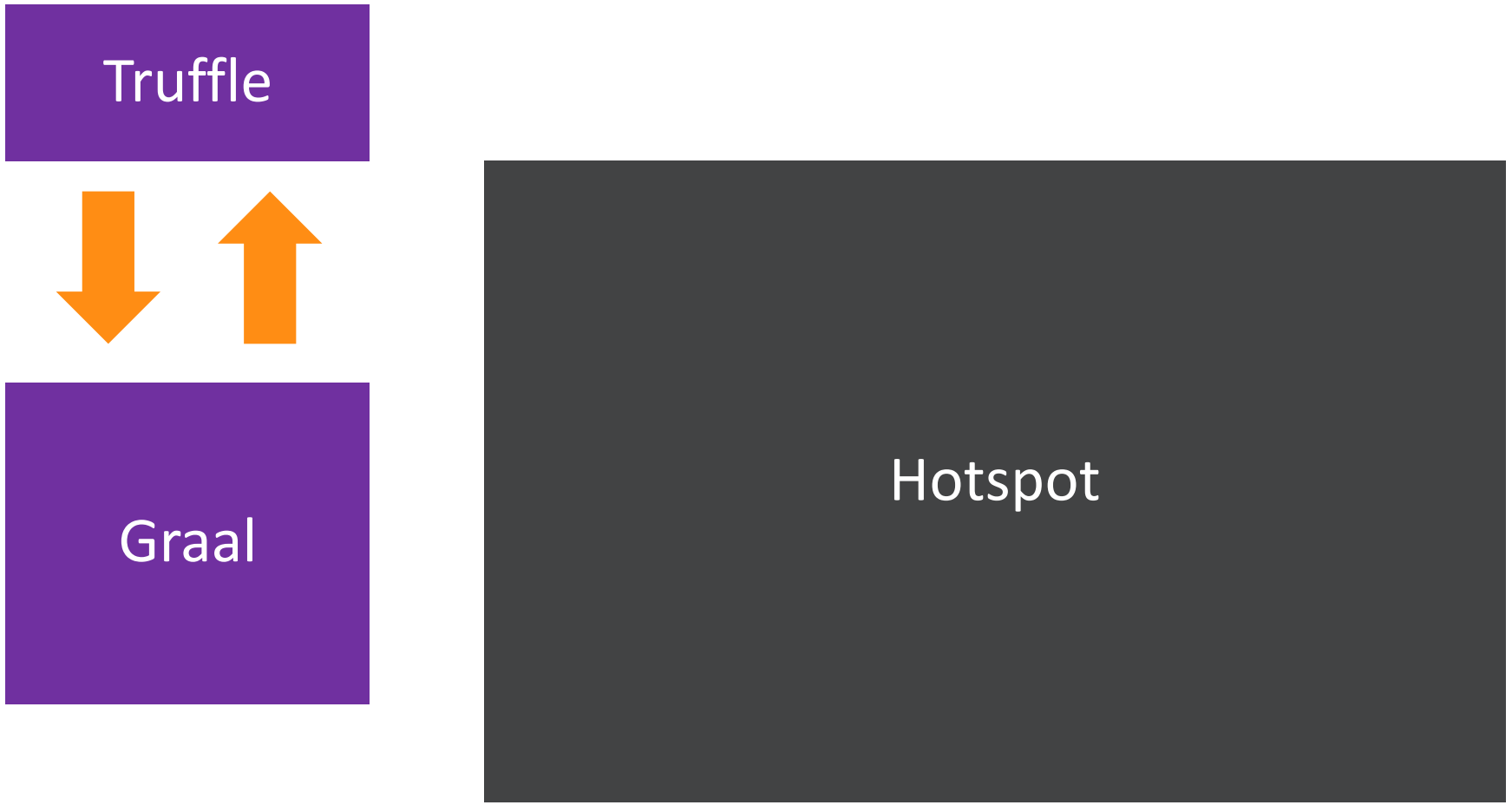
Hotspot











Slightly confusing terminology…

- Graal is a new JIT compiler for the JVM
- Graal VM is the JVM, with Graal, Truffle, and our languages bundled in it
- Truffle uses Graal on your behalf

How we do polyglot in GraalVM

```
Truffle::Interop.eval('application/language', source)

value = Truffle::Interop.import(name)

Truffle::Interop.export(name)
```

```
Interop.eval('application/language', source)

value = Interop.import(name)

Interop.export(name)
```

```ruby
puts Truffle::Interop.eval('application/javascript', '14 + 2')
# 16
```

JavaScript

Ruby

```ruby
puts Truffle::Interop.eval('application/javascript', '14 + 2')
# 16
```

```
Truffle::Interop.eval('application/javascript', "
  function add(a, b) {
    return a + b;
  }

  Interop.export('add', add.bind(this));
")

add = Truffle::Interop.import('add')

puts add.call(14, 2)
# 16
```

Ruby

```
Truffle::Interop.eval('application/javascript', "
  function add(a, b) {
    return a + b;
  }

  Interop.export('add', add.bind(this));
")

add = Truffle::Interop.import('add')

puts add.call(14, 2)
# 16
```

JavaScript

```
function add(a, b) {
  return a + b;
}

puts add(14, 2)
# 16
```

JavaScript

```
function add(a, b) {
  return a + b;
}
```

Ruby

```
puts add(14, 2)
# 16
```

```
function Point(x, y) {
  this.x = x;
  this.y = y;
}

function random_points(n) {
  points = [];
  for (i = 0; i < n; i++) {
    points[i] = new Point(Math.random(), Math.random())
  }
  return points;
}

points = random_points(100)

point = points[0]
puts point.x, point.y
# 0.642460680339328
# 0.116305386298814
```

JS

```
function Point(x, y) {
  this.x = x;
  this.y = y;
}

function random_points(n) {
  points = [];
  for (i = 0; i < n; i++) {
    points[i] = new Point(Math.random(), Math.random())
  }
  return points;
}
```

Ruby

```
points = random_points(100)

point = points[0]
puts point.x, point.y
# 0.642460680339328
# 0.116305386298814
```

Performance

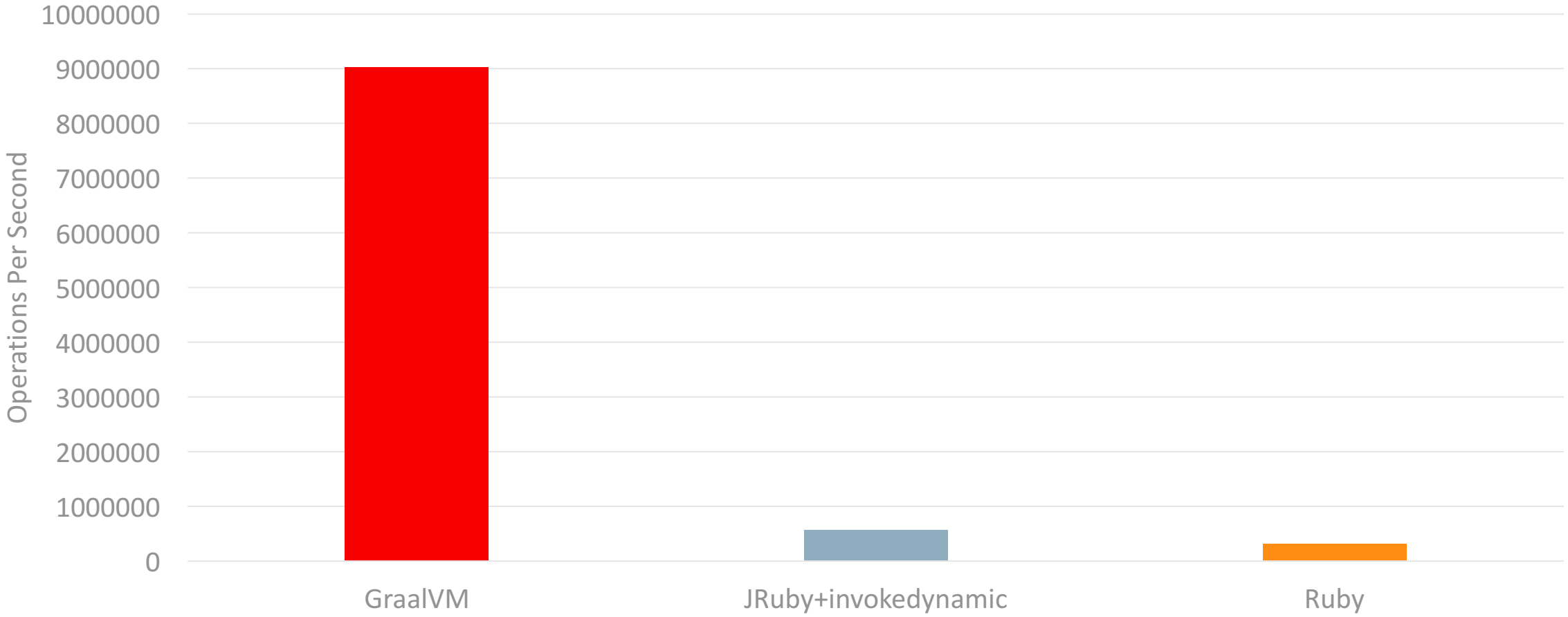
```ruby
def clamp(num, min, max)
  [min, num, max].sort[1]
end

def cmyk_to_rgb(c, m, y, k)
  Hash[{
    r: (65535 - (c * (255 - k) + (k << 8))) >> 8,
    g: (65535 - (m * (255 - k) + (k << 8))) >> 8,
    b: (65535 - (y * (255 - k) + (k << 8))) >> 8
  }.map { |k, v| [k, clamp(v, 0, 255)] }]
end

benchmark do
  cmyk_to_rgb(rand(255), rand(255), rand(255), rand(255))
end
```

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

def cmyk_to_rgb(c, m, y, k)
  Hash[{
    r: (65535 - (c * (255 - k) + (k << 8))) >> 8,
    g: (65535 - (m * (255 - k) + (k << 8))) >> 8,
    b: (65535 - (y * (255 - k) + (k << 8))) >> 8
  }.map { |k, v| [k, clamp(v, 0, 255)] }]
end

benchmark do
  cmyk_to_rgb(rand(255), rand(255), rand(255), rand(255))
end
```

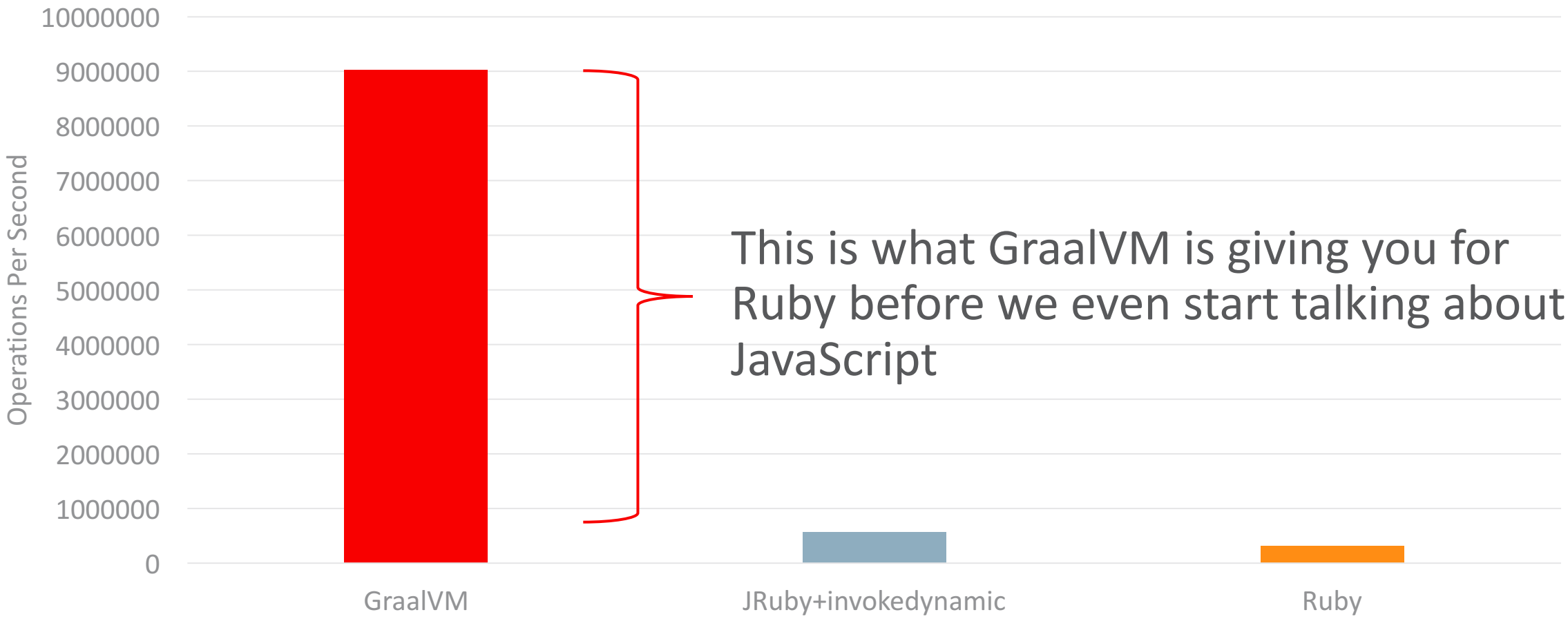
Warms up and then reports iterations per second

Random inputs stop the whole thing being totally optimised away

`clamp` in Pure Ruby



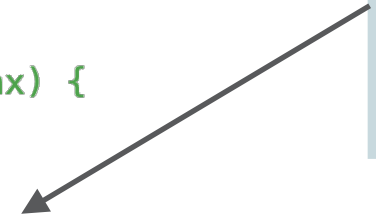
`clamp` in Pure Ruby



```ruby
require 'v8'

context = V8::Context.new

$clamp = context.eval("
  function clamp(num, min, max) {
    if (num < min) {
      return min;
    } else if (num > max) {
      return max;
    } else {
      return num;
    }
  }
  clamp;
")

def cmyk_to_rgb(c, m, y, k)
  Hash[{
    r: (65535 - (c * (255 - k) + (k << 8))) >> 8,
    g: (65535 - (m * (255 - k) + (k << 8))) >> 8,
    b: (65535 - (y * (255 - k) + (k << 8))) >> 8
  }.map { |k, v| [k, $clamp.call(v, 0, 255)] }]
end
```

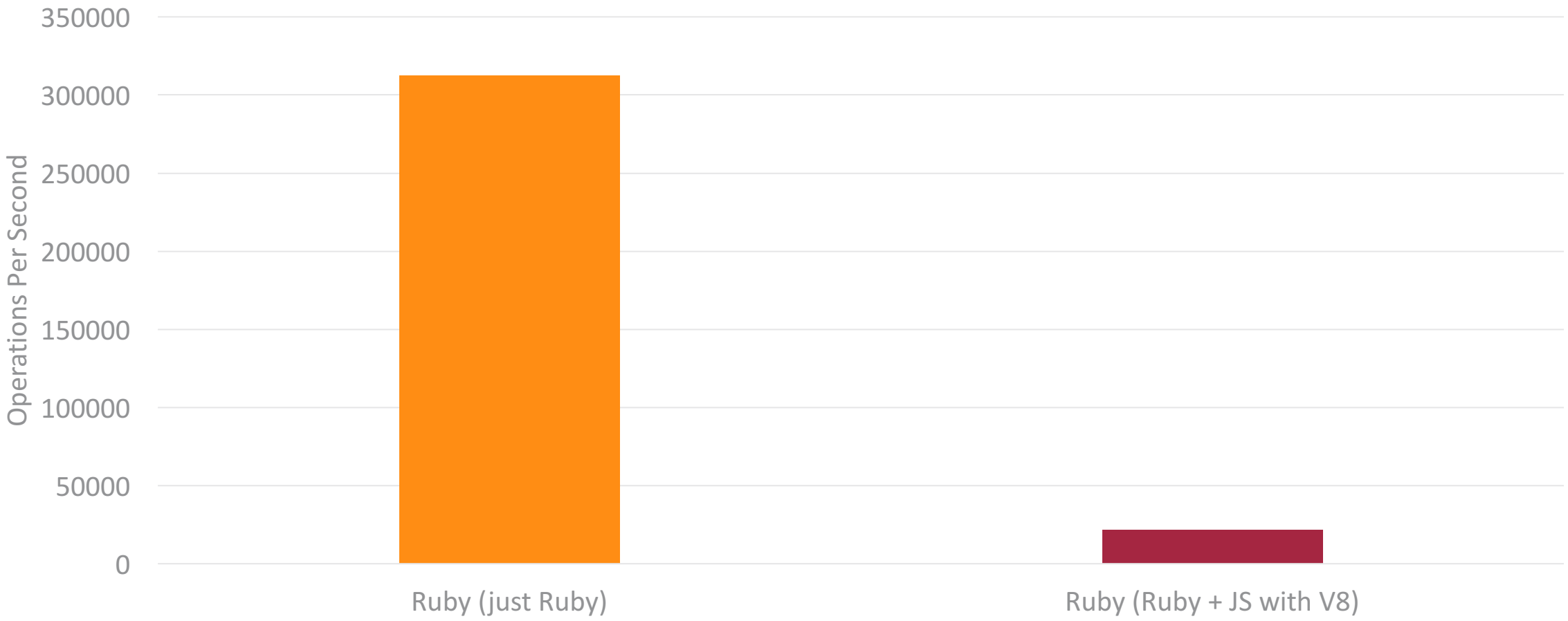
```ruby
require 'v8'

context = V8::Context.new

$clamp = context.eval("
  function clamp(num, min, max) {
    if (num < min) {
      return min;
    } else if (num > max) {
      return max;
    } else {
      return num;
    }
  }
  clamp;
")

def cmyk_to_rgb(c, m, y, k)
  Hash[{
    r: (65535 - (c * (255 - k) + (k << 8))) >> 8,
    g: (65535 - (m * (255 - k) + (k << 8))) >> 8,
    b: (65535 - (y * (255 - k) + (k << 8))) >> 8
  }.map { |k, v| [k, $clamp.call(v, 0, 255)] }]
end
```

Not only have we rewritten in JavaScript, but the JavaScript code is simpler than the Ruby



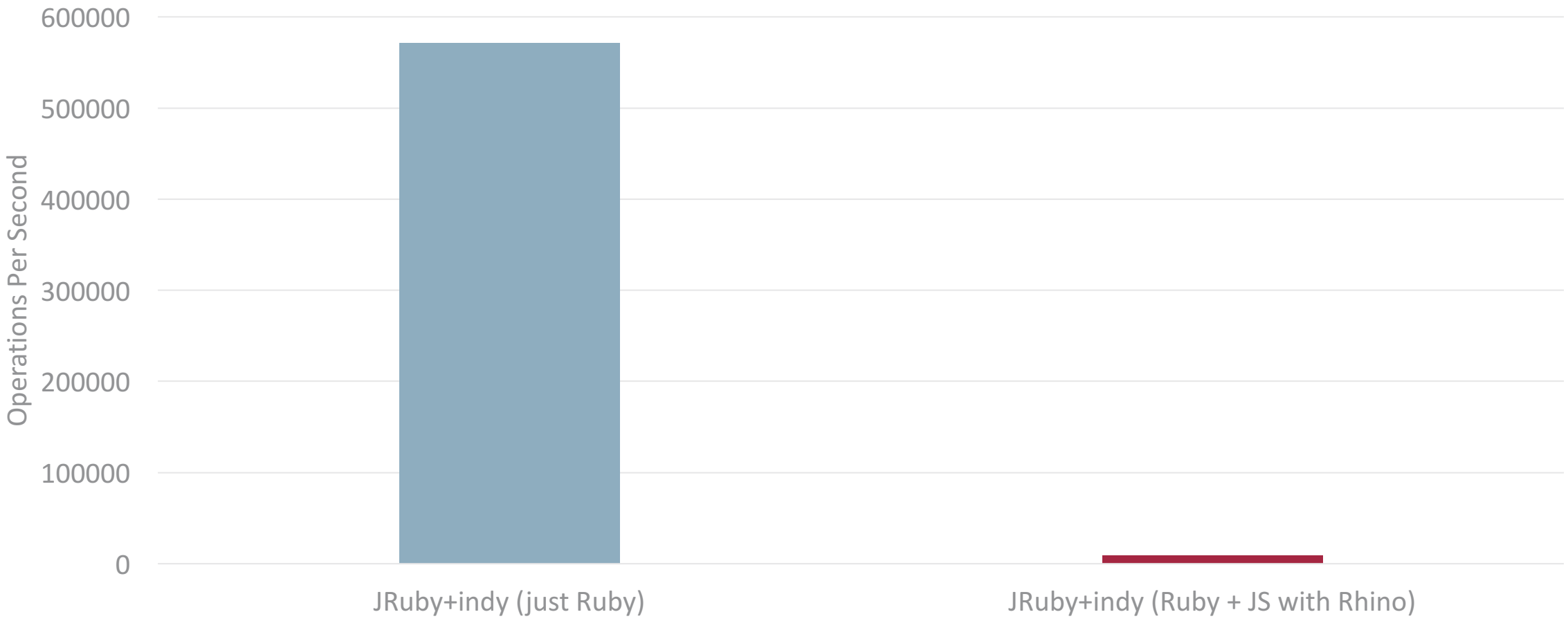
ORACLE

`clamp` in Ruby and JavaScript with V8



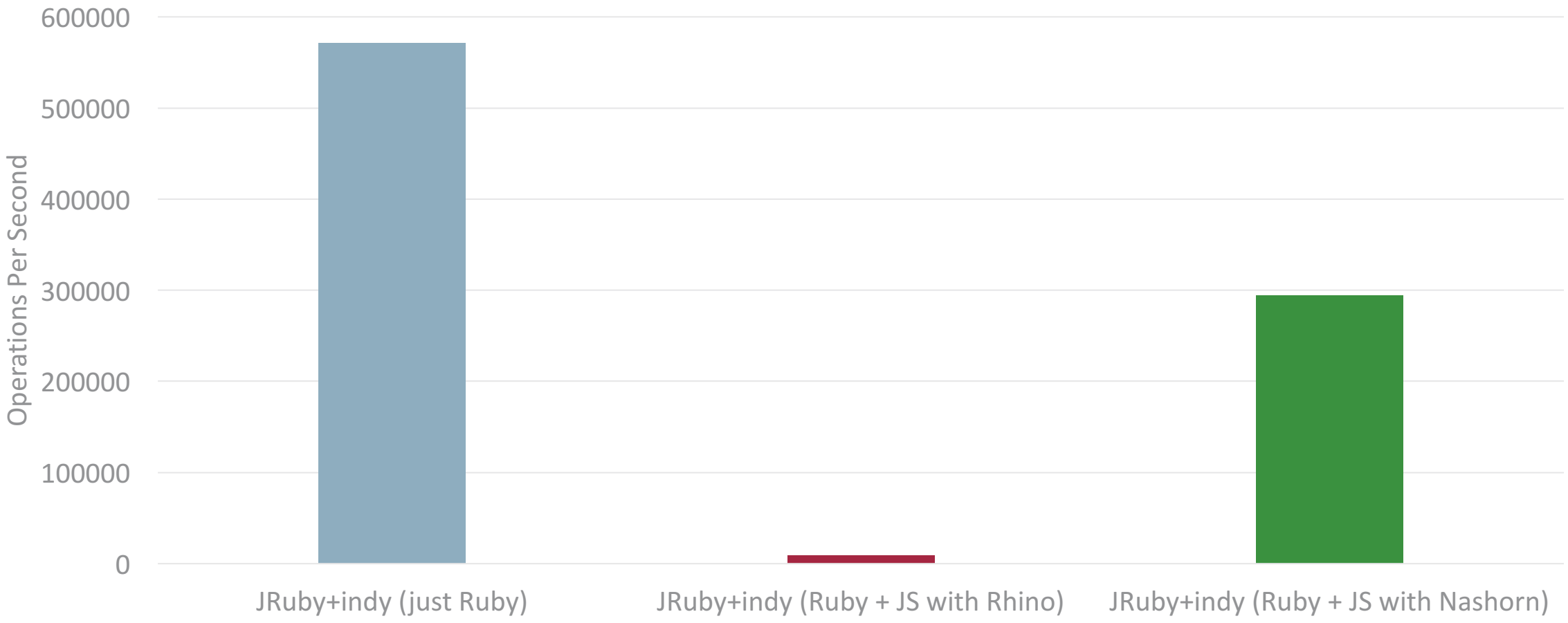
```
require 'rhino'

context = Rhino::Context.new
```

`clamp` in Ruby and JavaScript with JRuby and Rhino



```
factory = javax.script.ScriptEngineManager.new
engine = factory.getEngineByName 'nashorn'
bindings = engine.createBindings

$clamp = engine.eval("
  function clamp(num, min, max) {
    if (num < min) {
      return min;
    } else if (num > max) {
      return max;
    } else {
      return num;
    }
  }
", bindings)

def cmyk_to_rgb(c, m, y, k)
  Hash[{
    r: (65535 - (c * (255 - k) + (k << 8))) >> 8,
    g: (65535 - (m * (255 - k) + (k << 8))) >> 8,
    b: (65535 - (y * (255 - k) + (k << 8))) >> 8
  }.map { |k, v| [k, $clamp.call(v, 0, 255)] }]
end
```

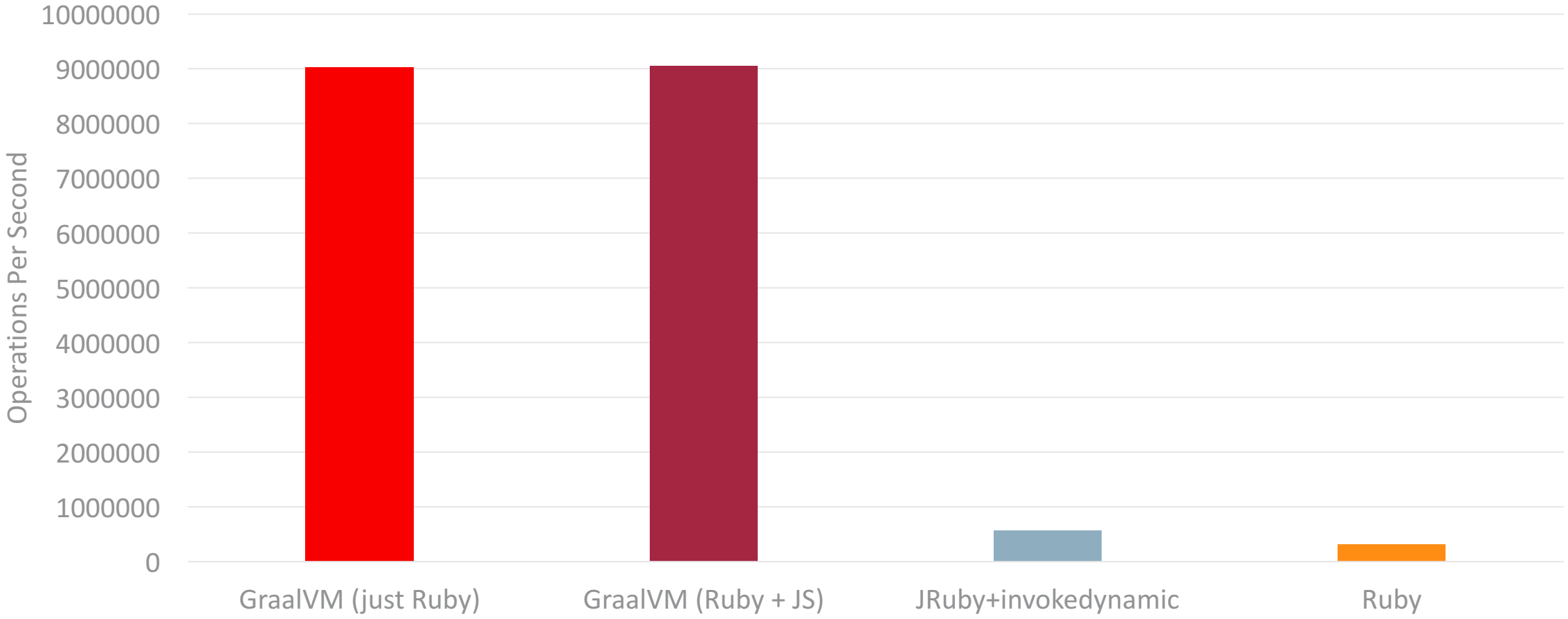
`clamp` in Ruby and JavaScript with JRuby and Nashorn



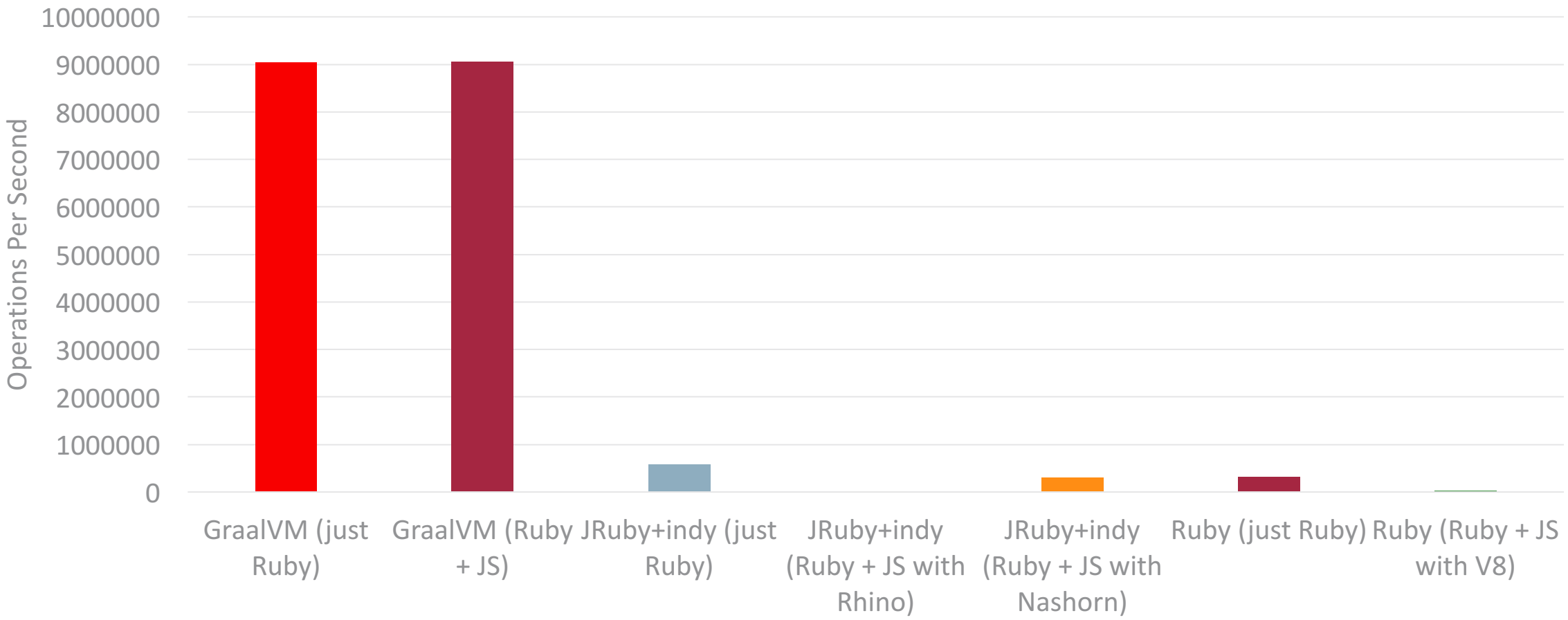
```
function clamp(num, min, max) {
  if (num < min) {
    return min;
  } else if (num > max) {
    return max;
  } else {
    return num;
  }
}
```

```
def cmyk_to_rgb(c, m, y, k)
  Hash[{
    r: (65535 - (c * (255 - k) + (k << 8))) >> 8,
    g: (65535 - (m * (255 - k) + (k << 8))) >> 8,
    b: (65535 - (y * (255 - k) + (k << 8))) >> 8
  }.map { |k, v| [k, clamp(v, 0, 255)] }]
end
```

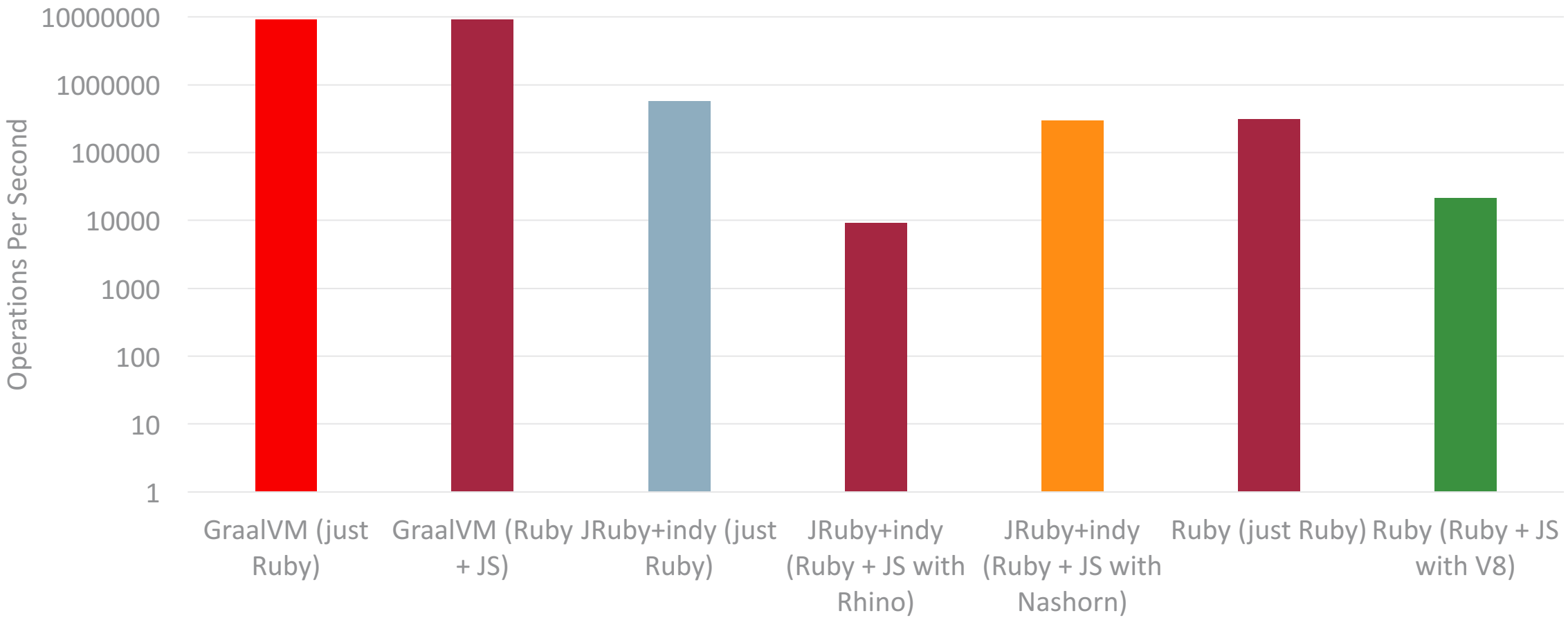
`clamp` in Ruby and JavaScript with GraalVM



`clamp` in all configurations

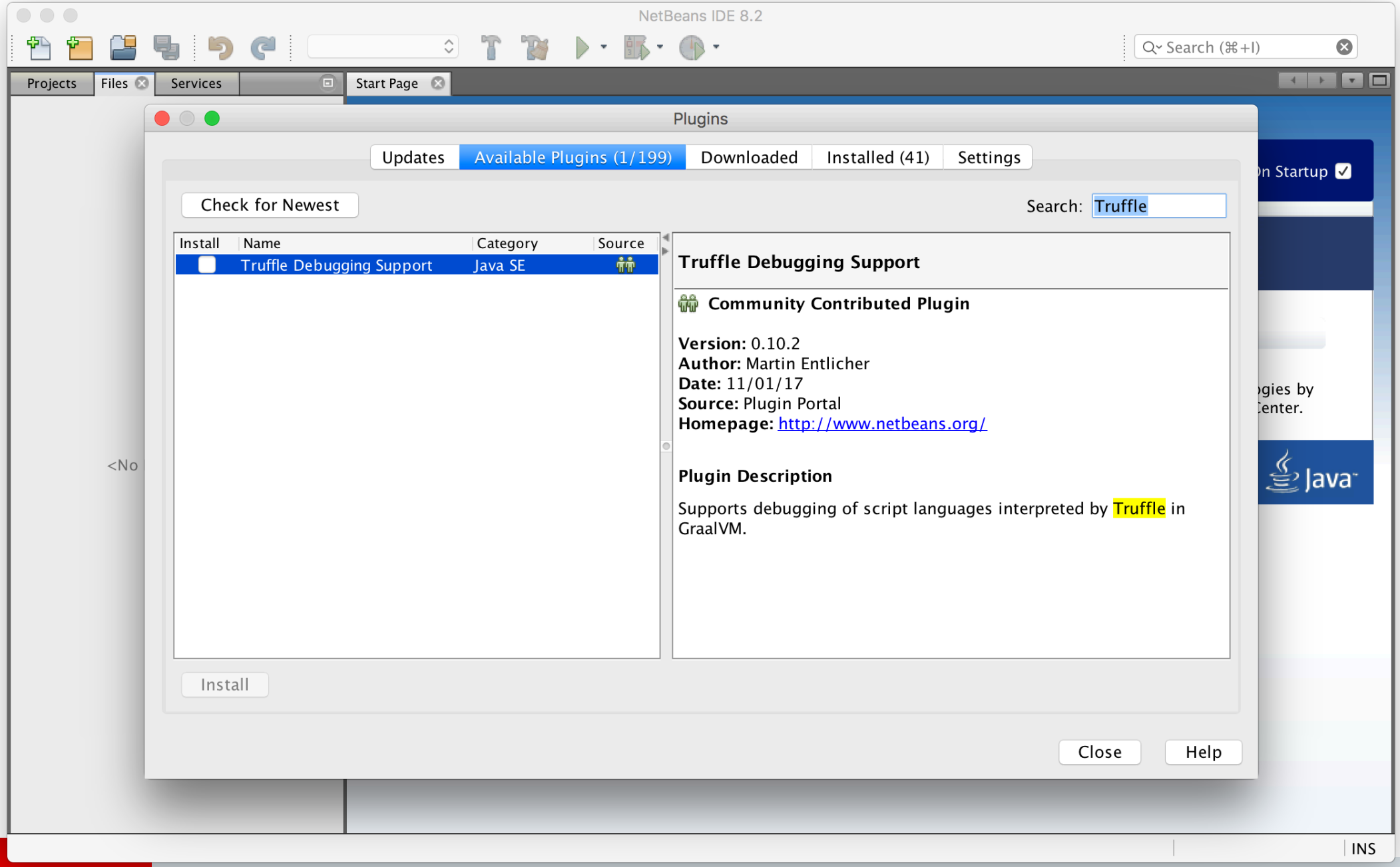


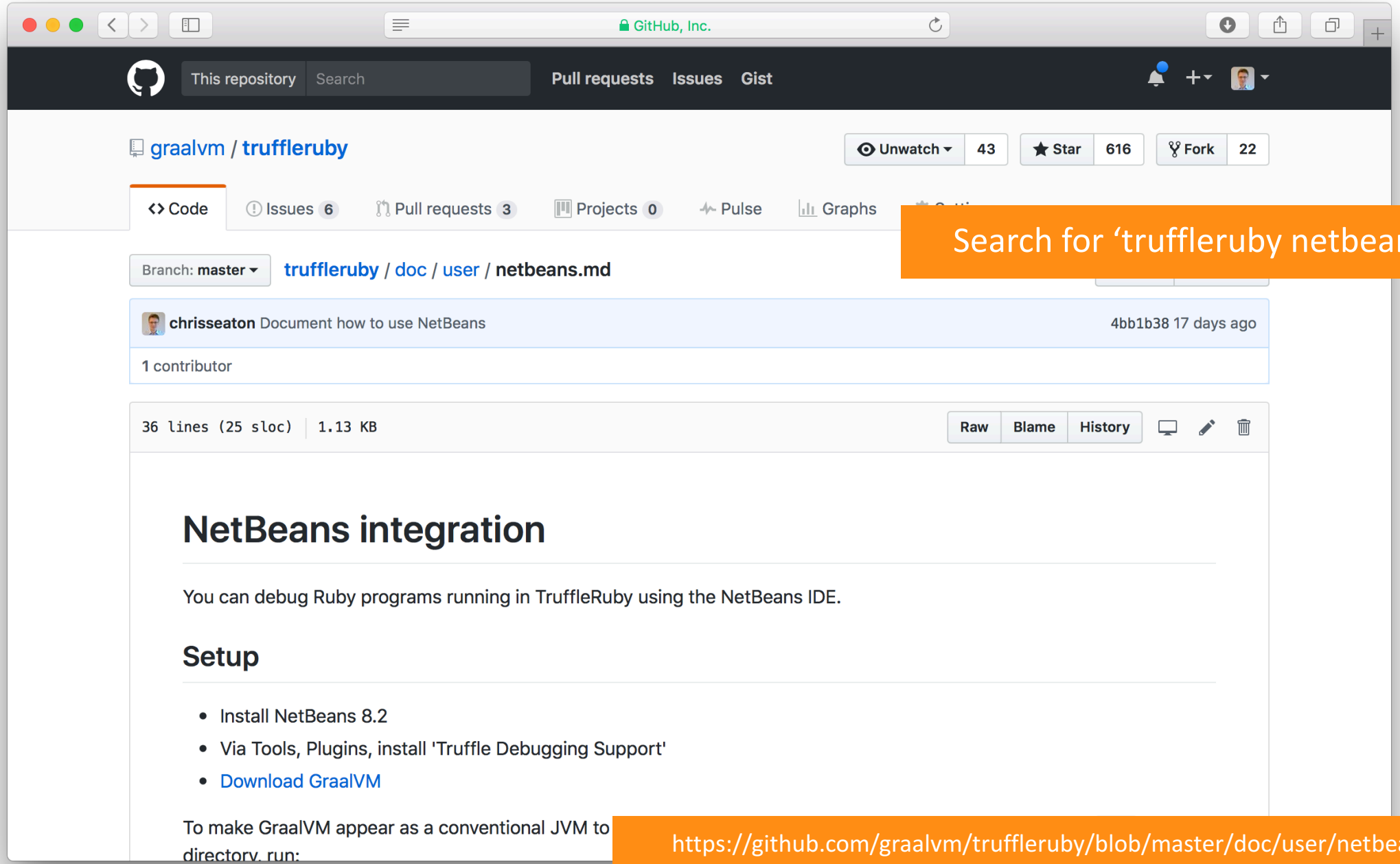
`clamp` in all configurations



GraalVM and NetBeans

NetBeans 8.2

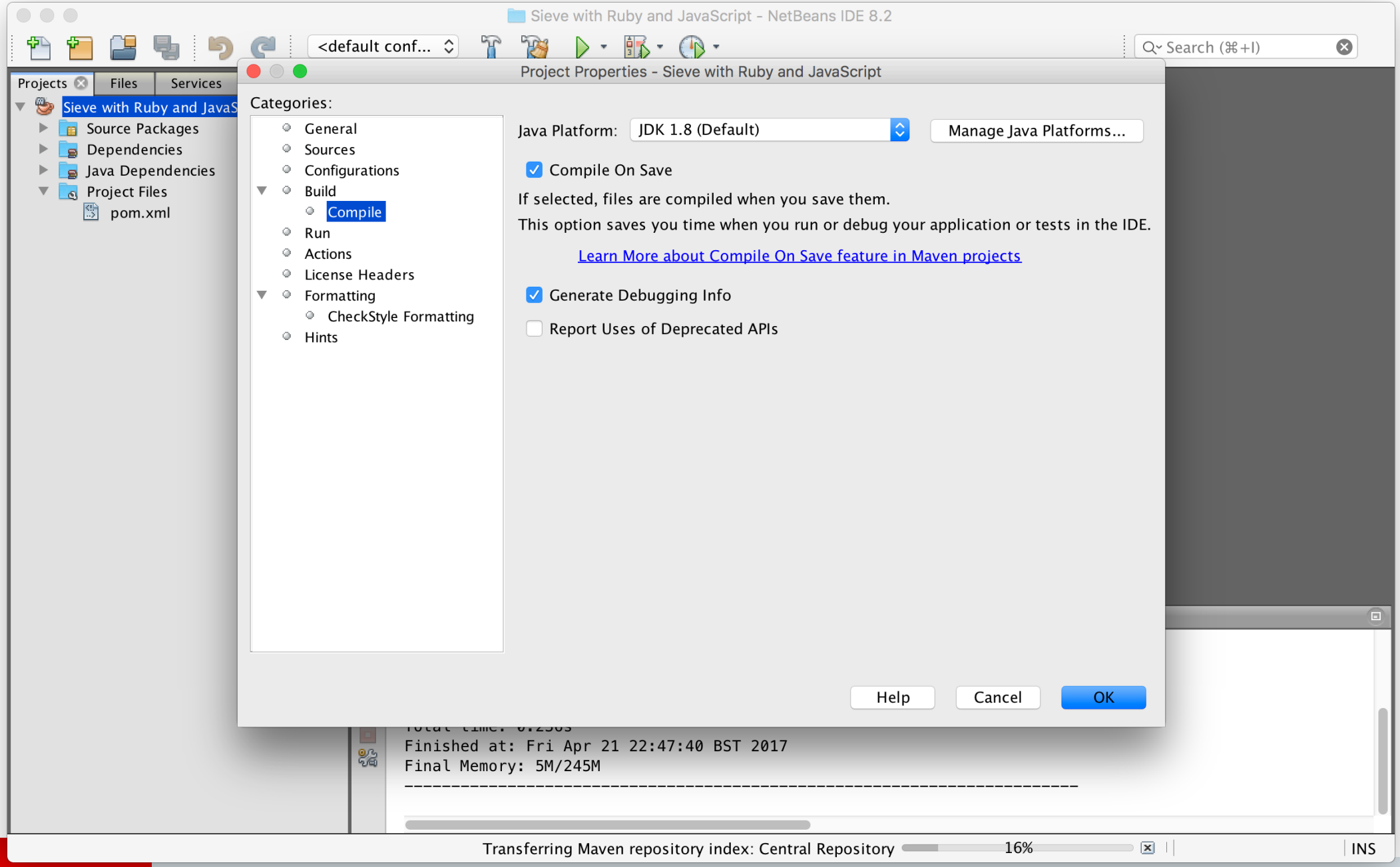


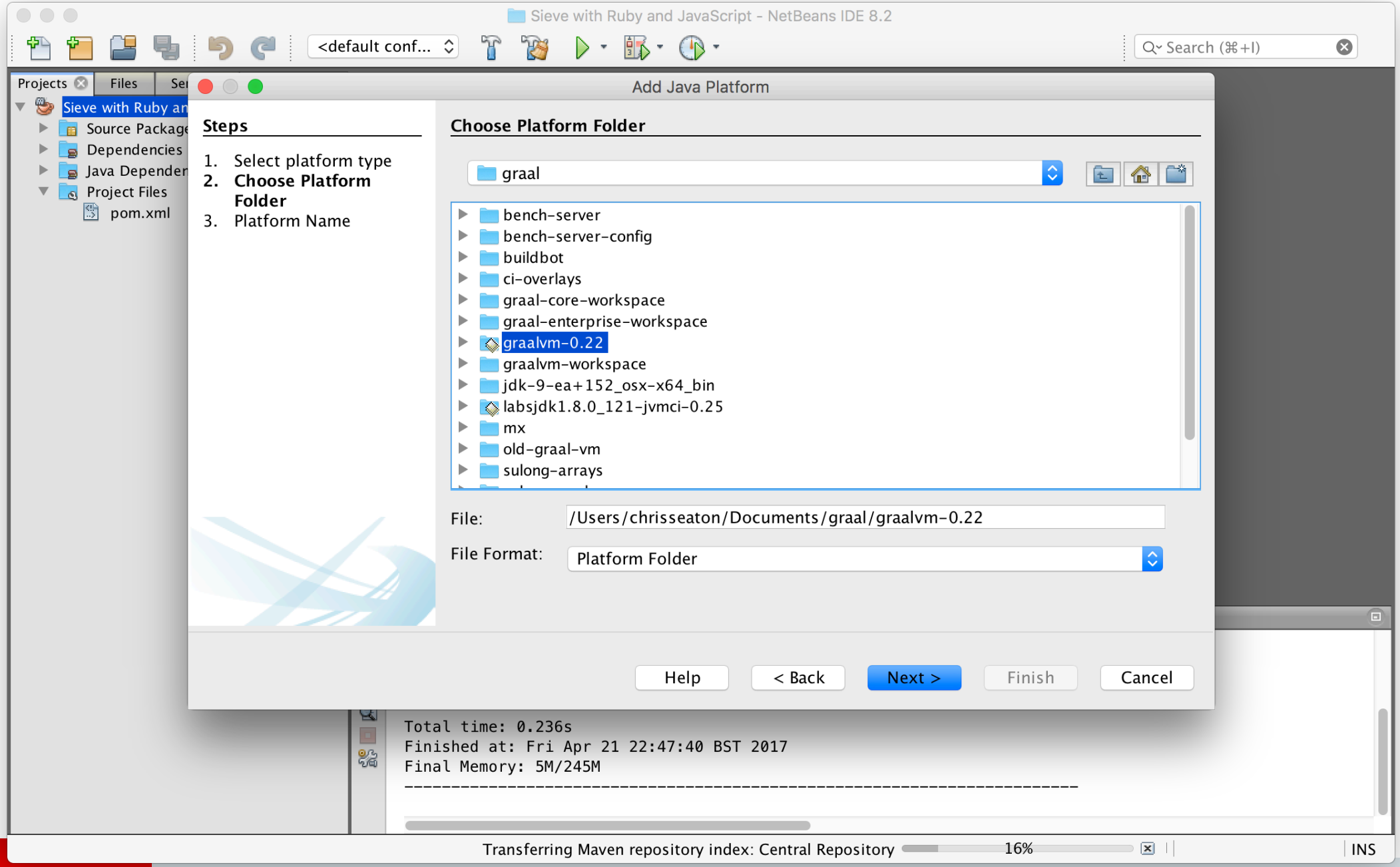


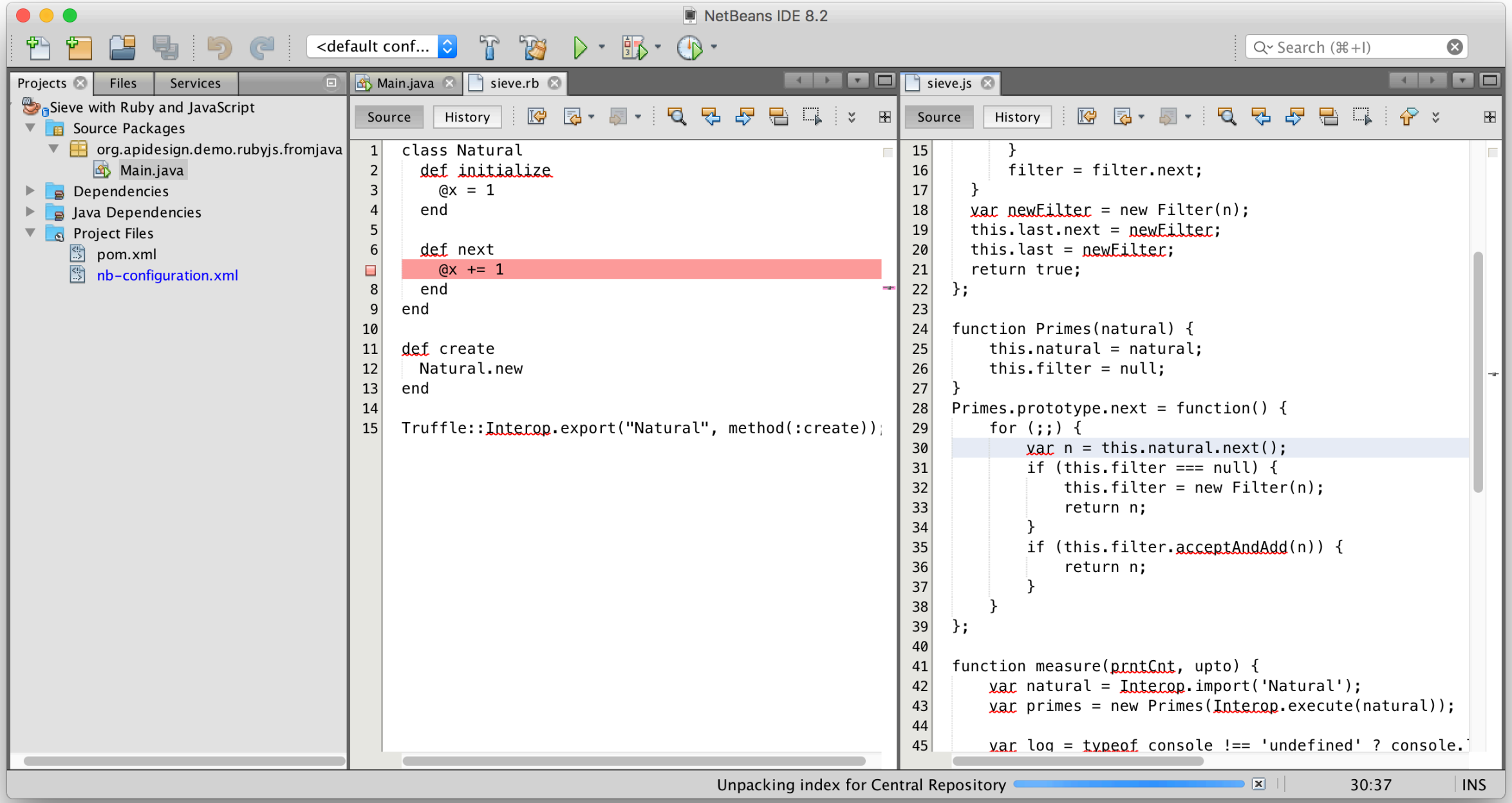
Search for ‘truffleruby netbeans’

https://github.com/graalvm/truffleruby/blob/master/doc/user/netbeans.md

git clone https://github.com/jtulach/sieve.git

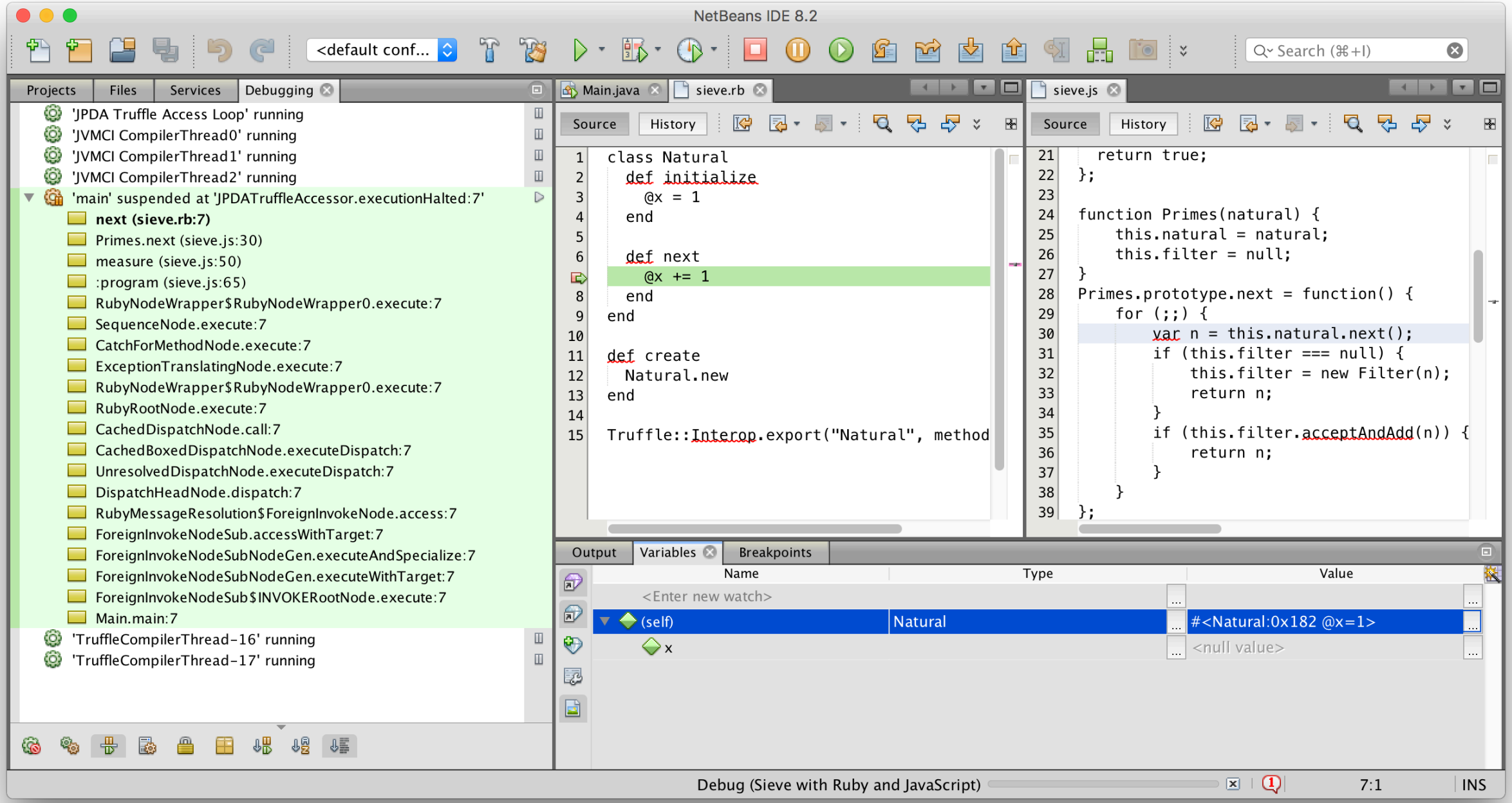






```ruby
class Natural
  def initialize
    @x = 1
  end

  def next
    @x += 1
  end
end

def create
  Natural.new
end

Truffle::Interop.export("Natural", method(:create));
```

```javascript
      }
      filter = filter.next;
  }
  var newFilter = new Filter(n);
  this.last.next = newFilter;
  this.last = newFilter;
  return true;
};

function Primes(natural) {
    this.natural = natural;
    this.filter = null;
}
Primes.prototype.next = function() {
    for (;;) {
        var n = this.natural.next();
        if (this.filter === null) {
            this.filter = new Filter(n);
            return n;
        }
        if (this.filter.acceptAndAdd(n)) {
            return n;
        }
    }
};

function measure(prntCnt, upto) {
    var natural = Interop.import('Natural');
    var primes = new Primes(Interop.execute(natural));
```

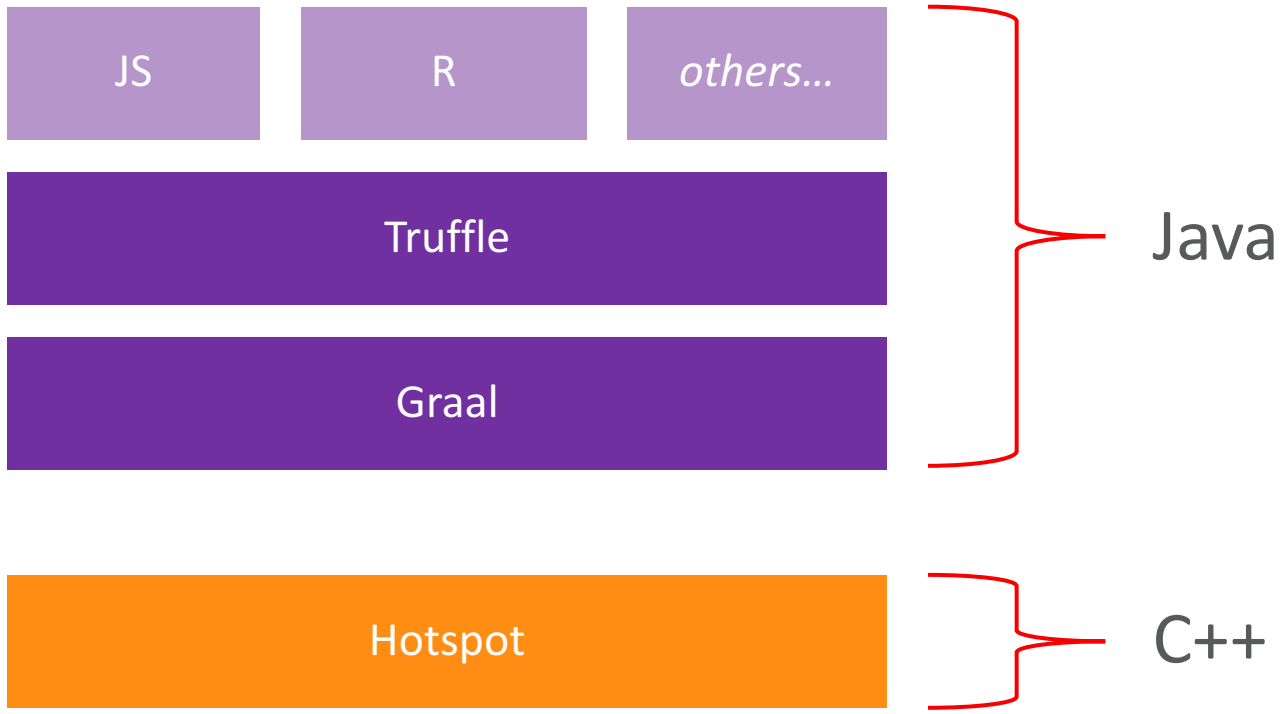


How to use GraalVM

GraalVM – everything in one package today

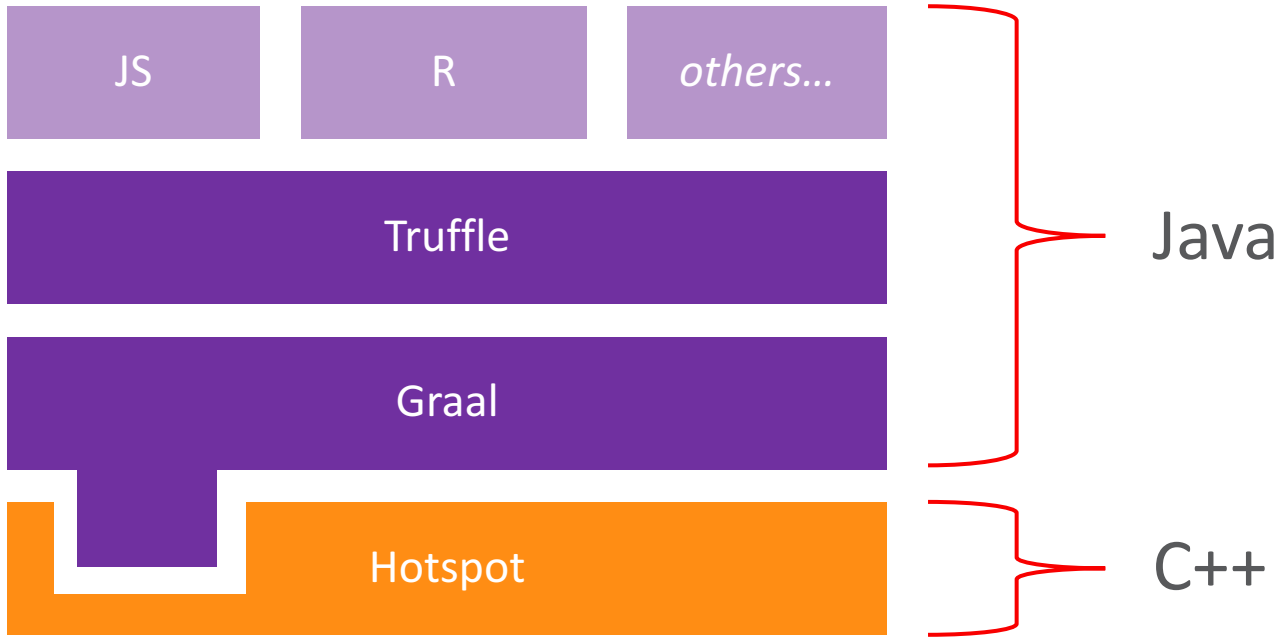
- Includes:
  - JVM (RE or DK)
  - Java
  - JavaScript
  - Ruby
  - R
  - More in the future
- Binary tarball release
- Mac or Linux

Java 9 – runs on an unmodified JVM

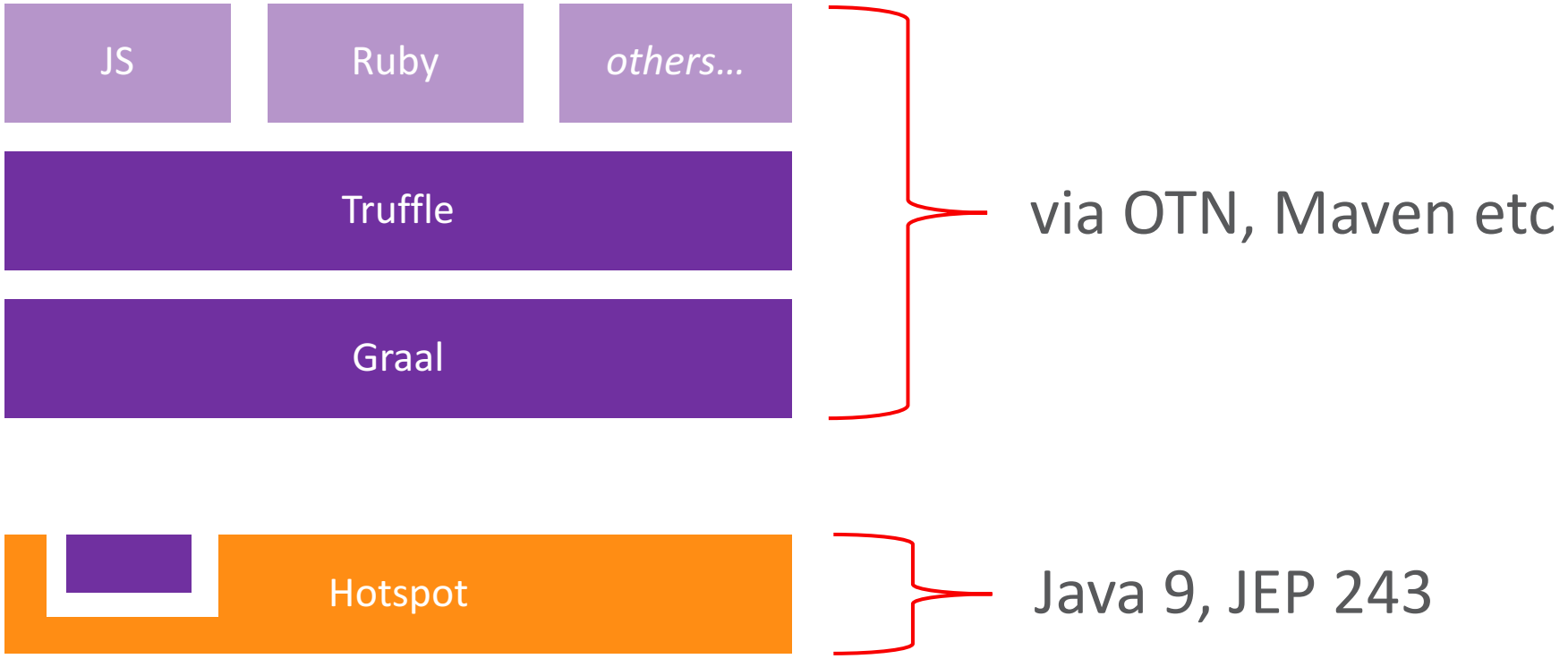


Java 9 – runs on an unmodified JVM

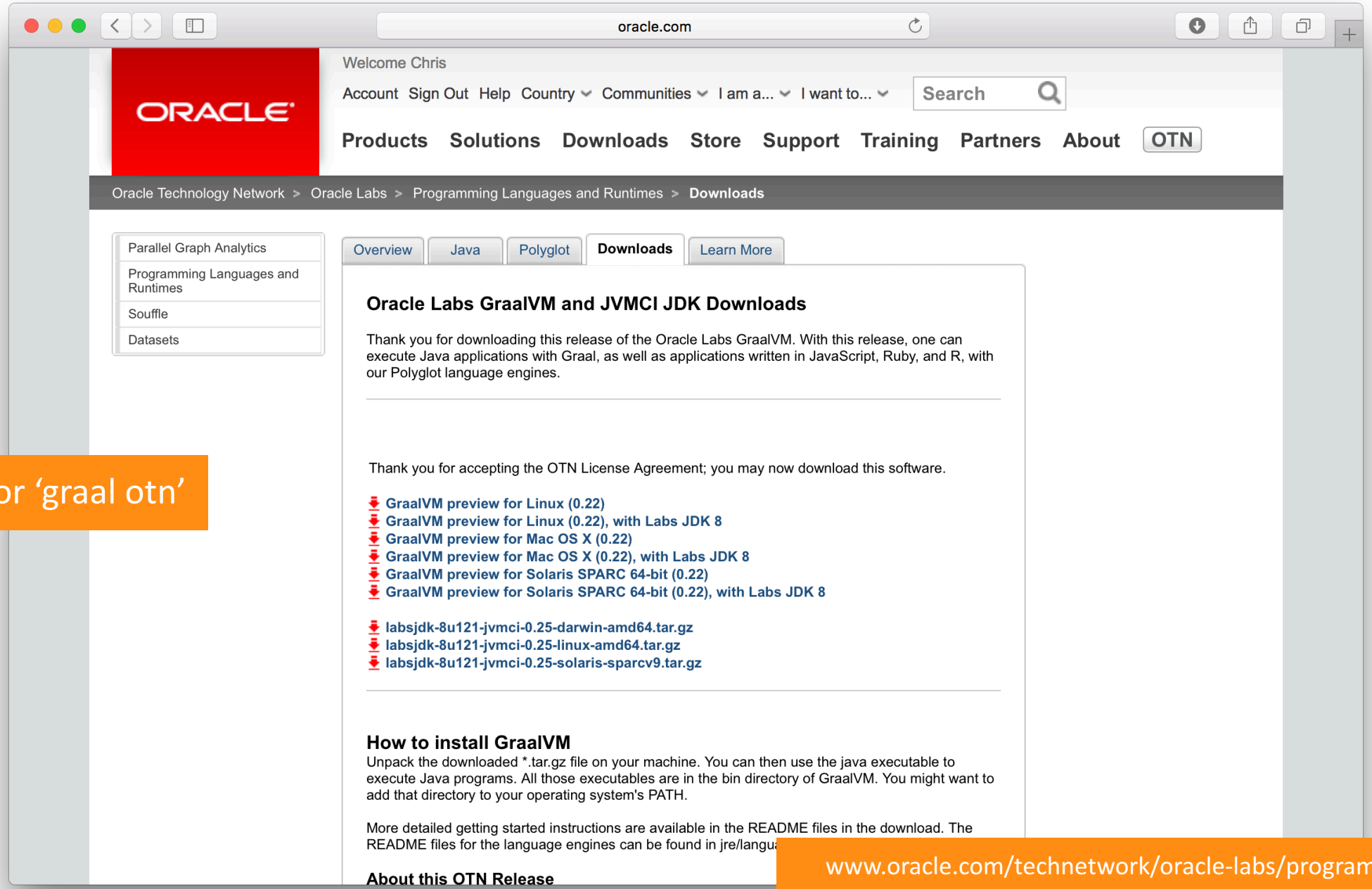
JVMCI
(JVM Compiler Interface)



Java 9 – runs on an unmodified JVM

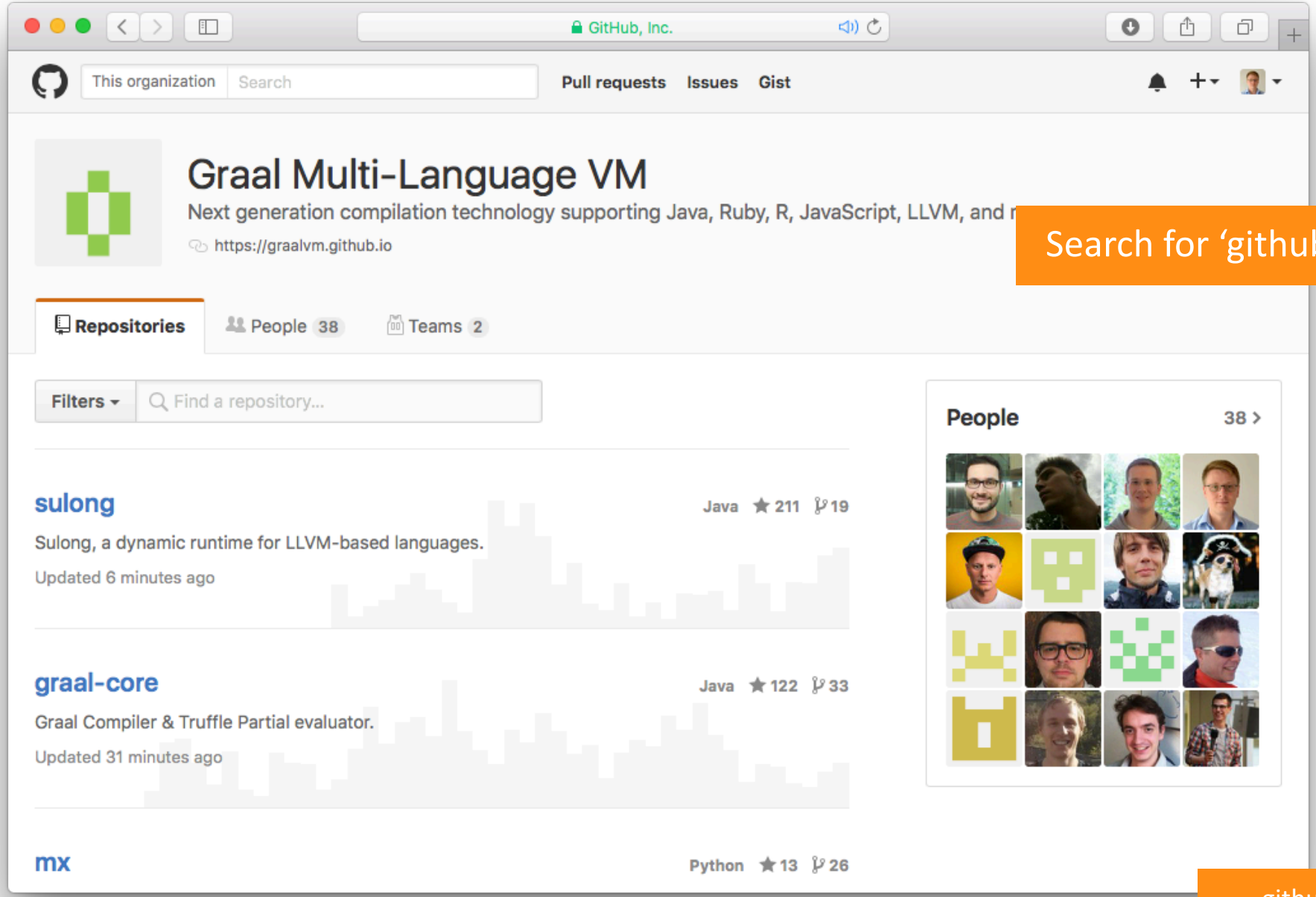


Where to find more information



Search for ‘graal otn’

www.oracle.com/technetwork/oracle-labs/program-languages



Search for ‘github graalvm’

github.com/graalvm

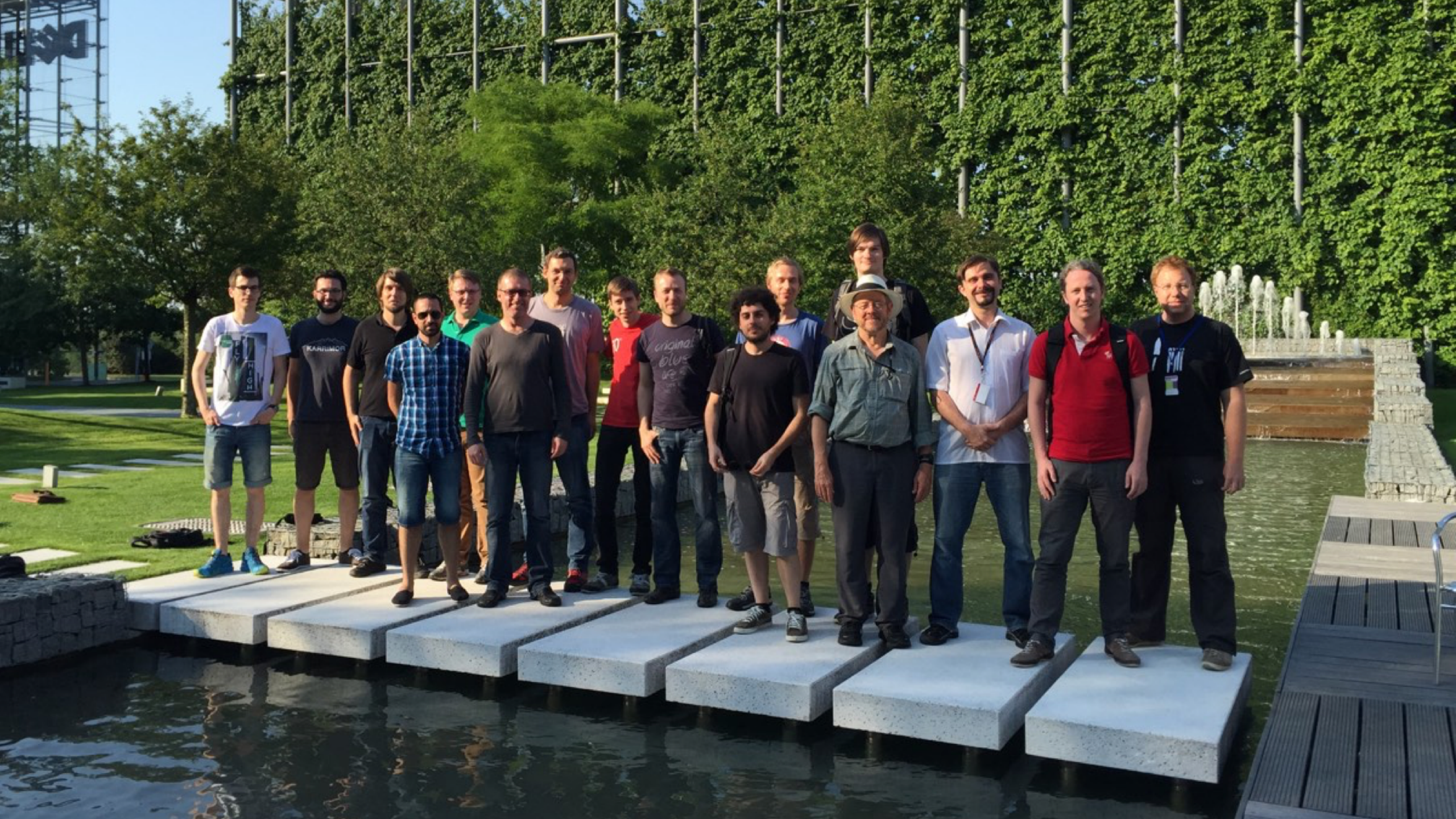
ORACLE

@chrisgseaton

github.com/graalvm

gitter.im/graalvm/graal-core

Search ‘otn graalvm’



Acknowledgements

Oracle
Danilo Ansaloni
Stefan Anzinger
Cosmin Basca
Daniele Bonetta
Matthias Brantner
Petr Chalupa
Jürgen Christ
Laurent Daynès
Gilles Duboscq
Martin Entlicher
Brandon Fish
Bastian Hossbach
Christian Humer
Mick Jordan
Vojin Jovanovic
Peter Kessler
David Leopoldseder
Kevin Menard
Jakub Podlešák
Aleksandar Prokopec
Tom Rodriguez

Oracle (continued)
Roland Schatz
Chris Seaton
Doug Simon
Štěpán Šindelář
Zbyněk Šlajchrt
Lukas Stadler
Codrut Stancu
Jan Štola
Jaroslav Tulach
Michael Van De Vanter
Adam Welc
Christian Wimmer
Christian Wirth
Paul Wögerer
Mario Wolczko
Andreas Wöß
Thomas Würthinger

Oracle Interns
Brian Belleville
Miguel Garcia
Shams Imam
Alexey Karyakin
Stephen Kell
Andreas Kunft
Volker Lanting
Gero Leinemann
Julian Lettner
Joe Nash
David Piorkowski
Gregor Richards
Robert Seilbeck
Rifat Shariyar

Alumni
Erik Eckstein
Michael Haupt
Christos Kotselidis
Hyunjin Lee
David Leibs
Chris Thalinger
Till Westmann

JKU Linz
Prof. Hanspeter Mössenböck
Benoit Daloze
Josef Eisl
Thomas Feichtinger
Matthias Grimmer
Christian Häubl
Josef Haider
Christian Huber
Stefan Marr
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
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

University of Lugano, Switzerland
Prof. Walter Binder
Sun Haiyang
Yudi Zheng

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


ORACLE®