

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



stackoverflow

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?



stackoverflow

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-all/](#) ▾

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](#) ▾

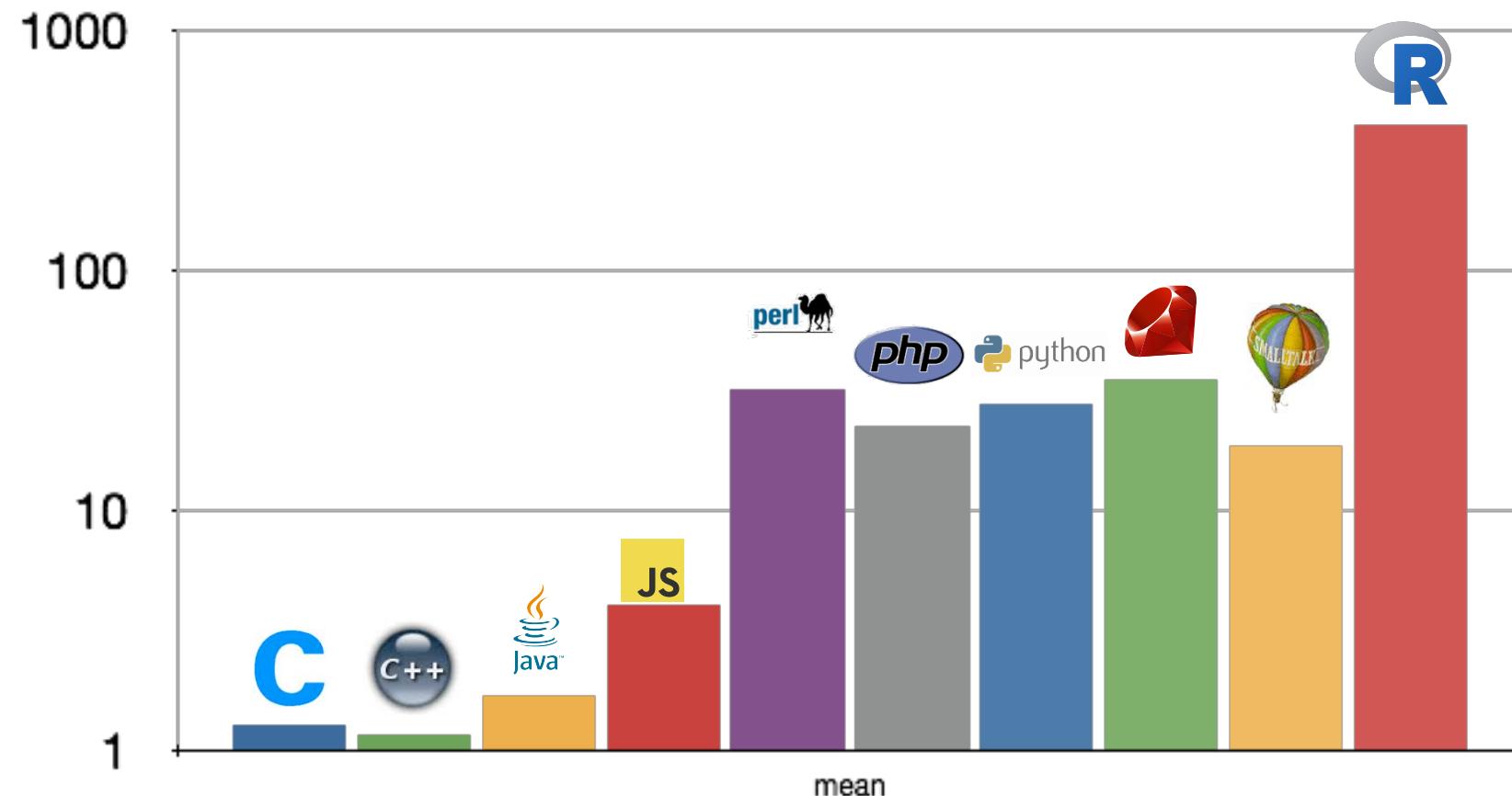
Oct 10, 2005 - **One Language to Rule Them All - Java**. For a long time I'd been hoping to add a scripting language to LibQ, to use in any of my (or other ...

[Dart : one language to rule them all - MixIT 2013 - Slideshare](#)

[fr.slideshare.net/sdeleuze/dart-mixit2013en](#) ▾

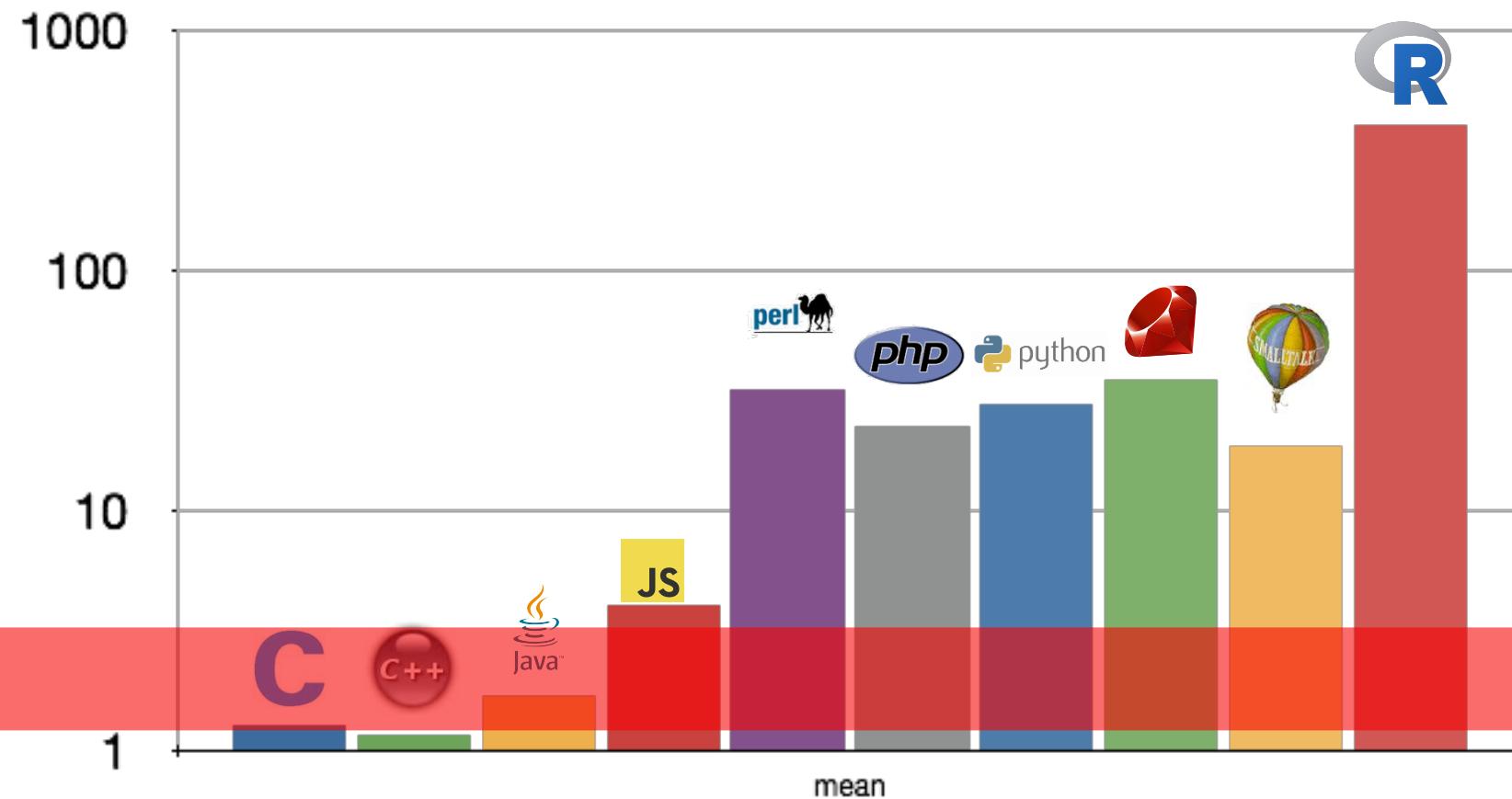
DartSébastien Deleuze - @sdeleuzeMix-IT 2013One language to rule them all ...

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

How it should be

Prototype a new language

Parser and language work to build syntax tree (AST), AST Interpreter

Write a “real” VM

In C/C++, still using AST interpreter, spend a lot of time implementing runtime system, GC, ...

People start using it

People complain about performance

Define a bytecode format and write bytecode interpreter

Performance is still bad

Write a JIT compiler
Improve the garbage collector

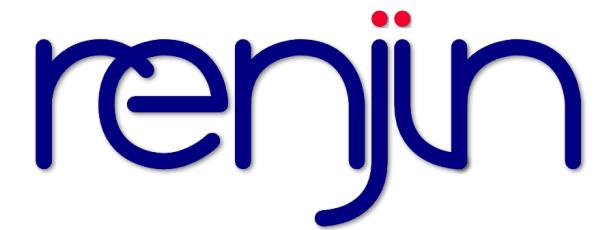
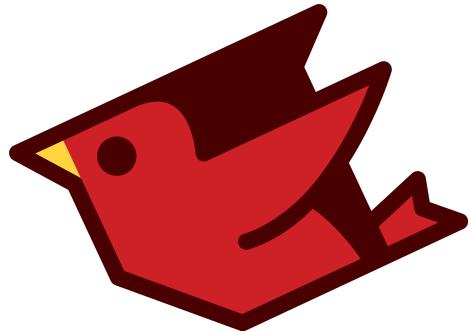
Prototype a new language in Java

Parser and language work to build syntax tree (AST)
Execute using AST interpreter

People start using it

And it is already fast

The Graal Approach



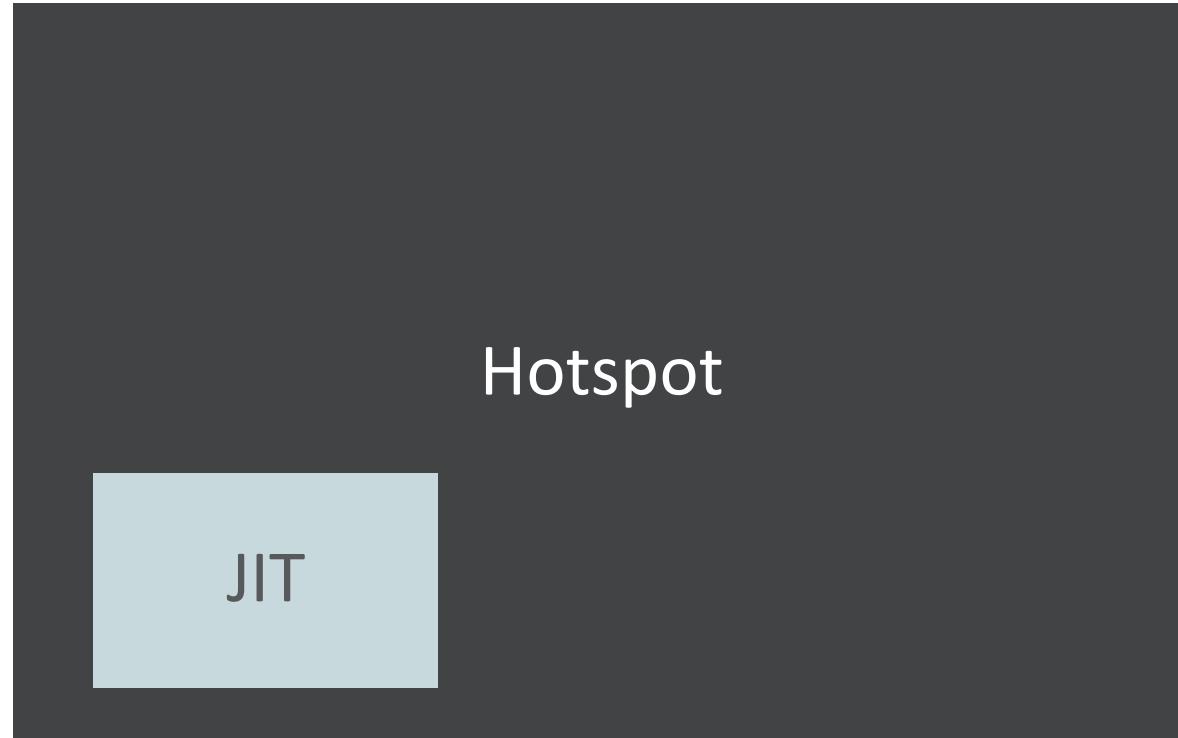
Conventional JVM implementations of languages work by emitting
JVM bytecode – the same thing that the Java compiler does

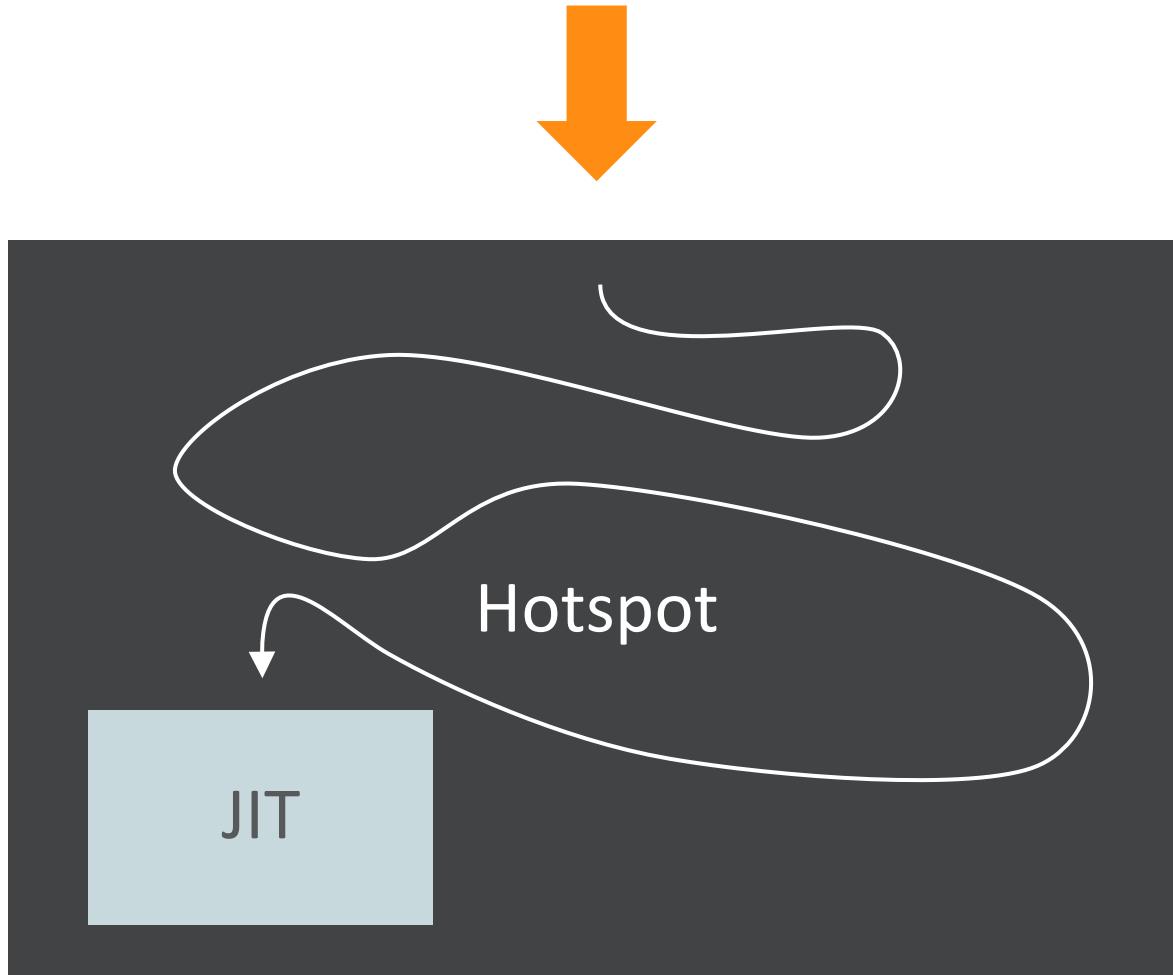


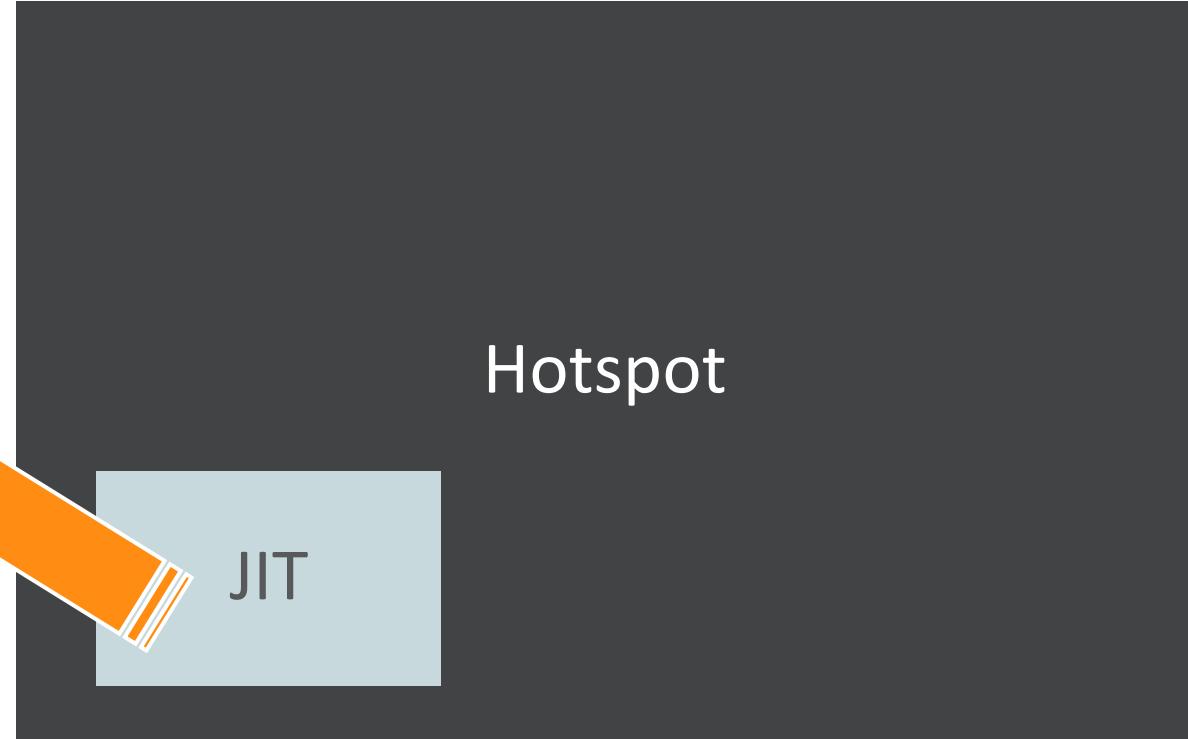
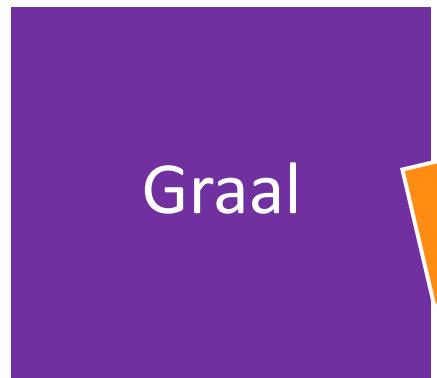
Hotspot



Hotspot







Graal

Hotspot

Truffle



Graal

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)
```

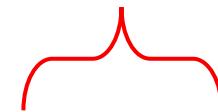
```
Interop.export(name)
```

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

Ruby

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

JavaScript



```
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

```
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

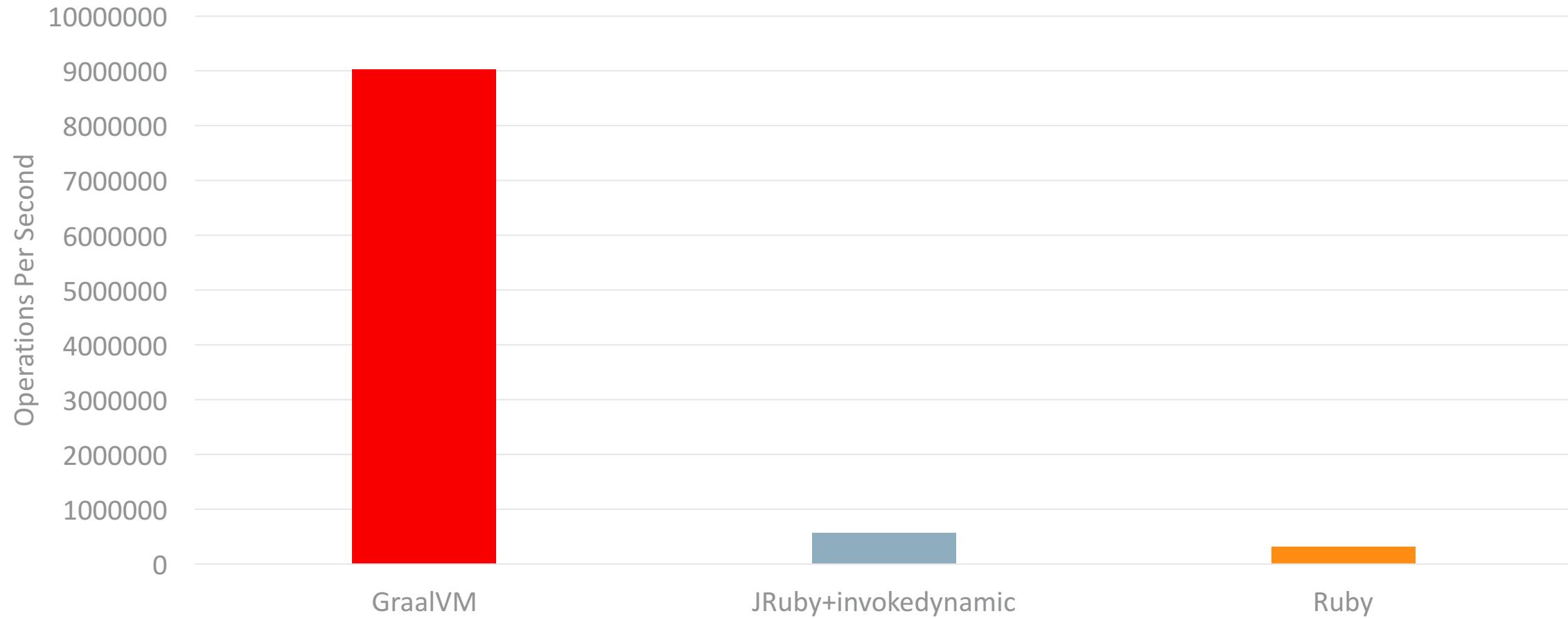
```
def clamp(num, min, max)
  [min, num, max].sort[1]
end

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

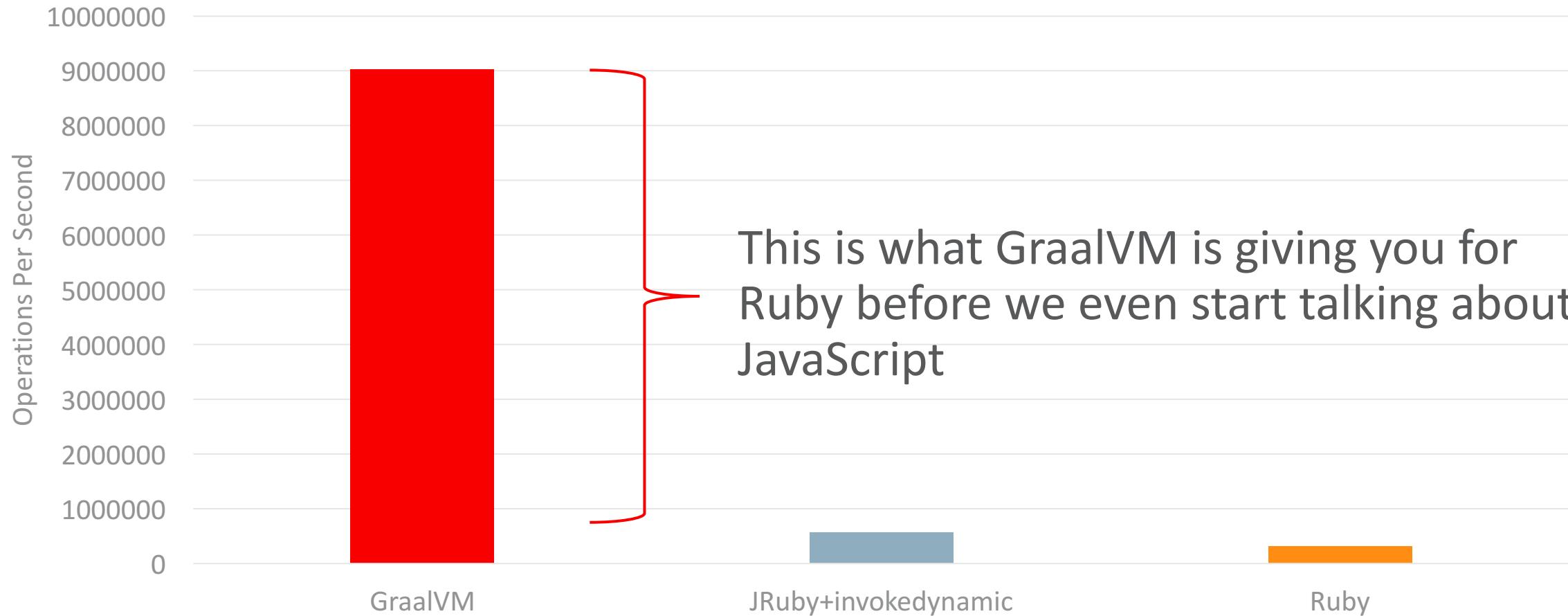
benchmark do
  cmyk_to_rgb(rand(255), rand(255), rand(255), rand(255))
end
```

Random inputs stop the
whole thing being totally
optimised away

clamp in Pure Ruby



clamp in Pure 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
```

```
require 'v8'

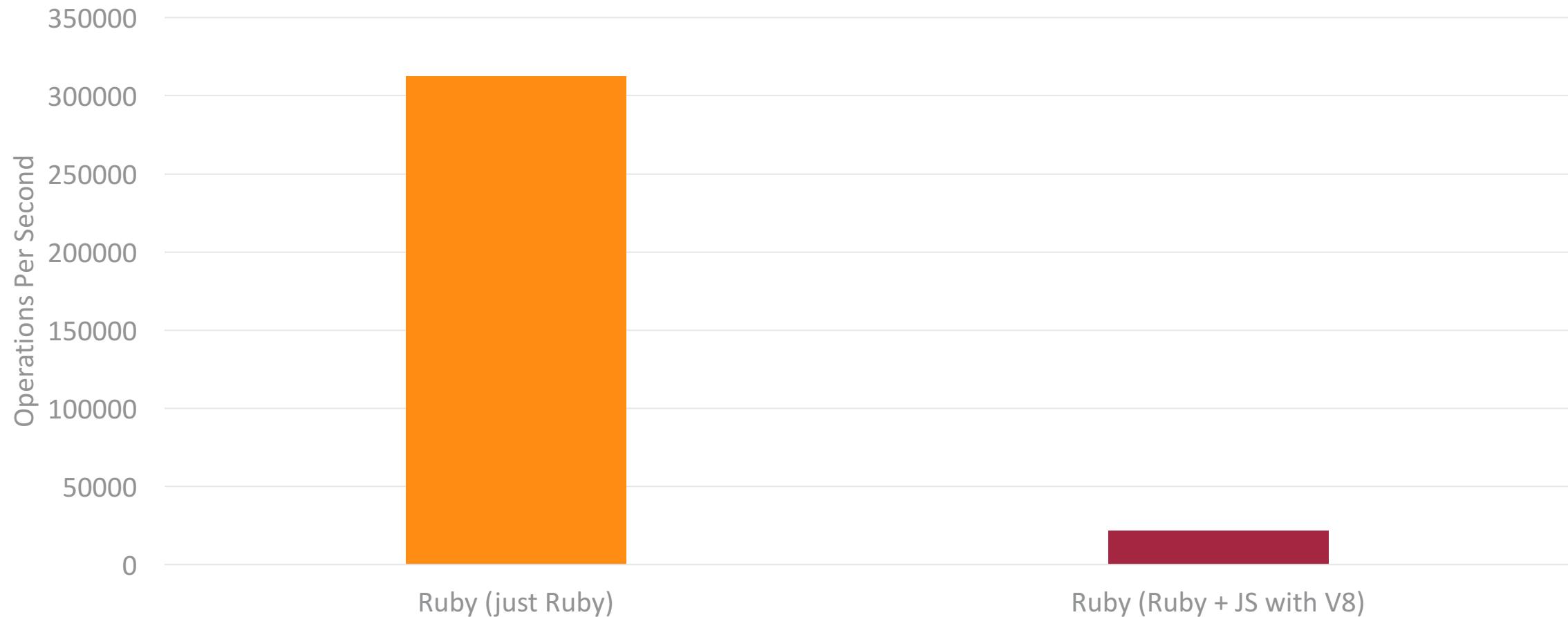
context = V8::Context.new

$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
```

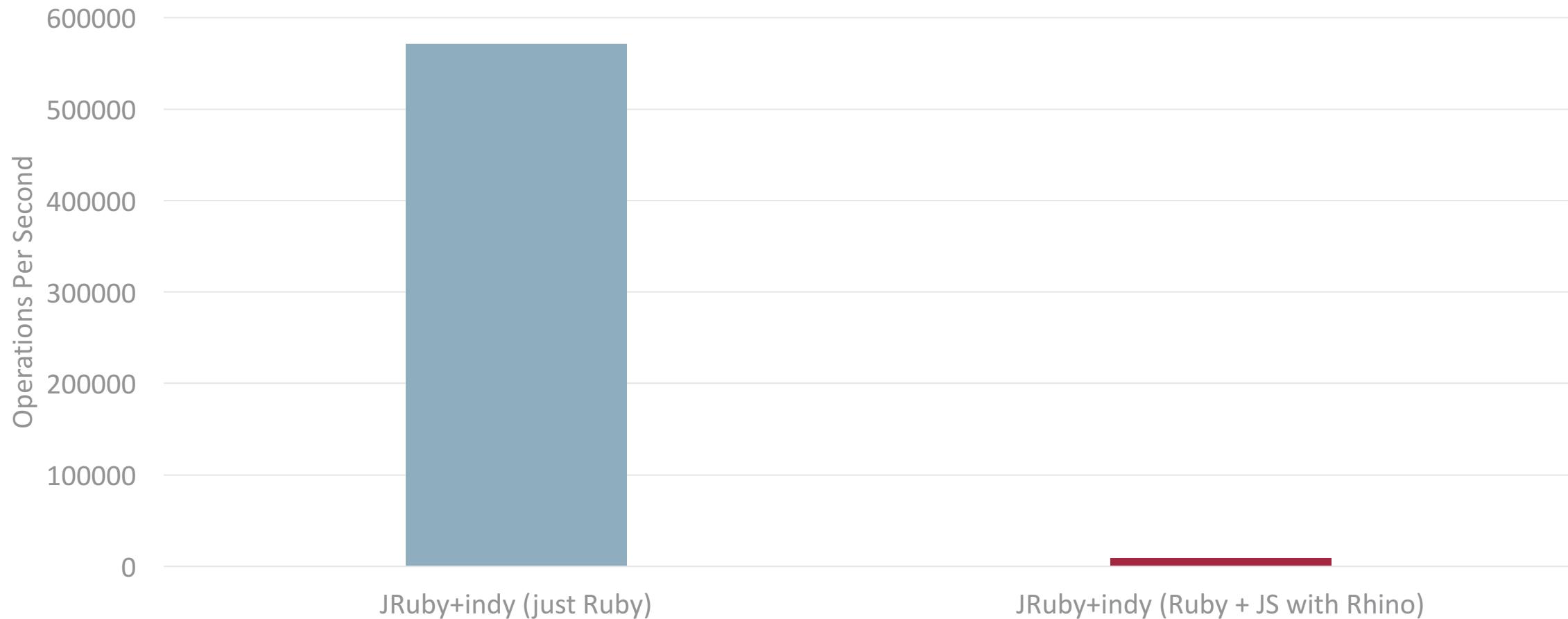
Not only have we rewritten
in JavaScript, but the
JavaScript code is simpler
than the Ruby

clamp in Ruby and JavaScript with V8



```
require 'rhino'  
  
context = Rhino::Context.new
```

clamp in Ruby and JavaScript with JRuby and Rhino

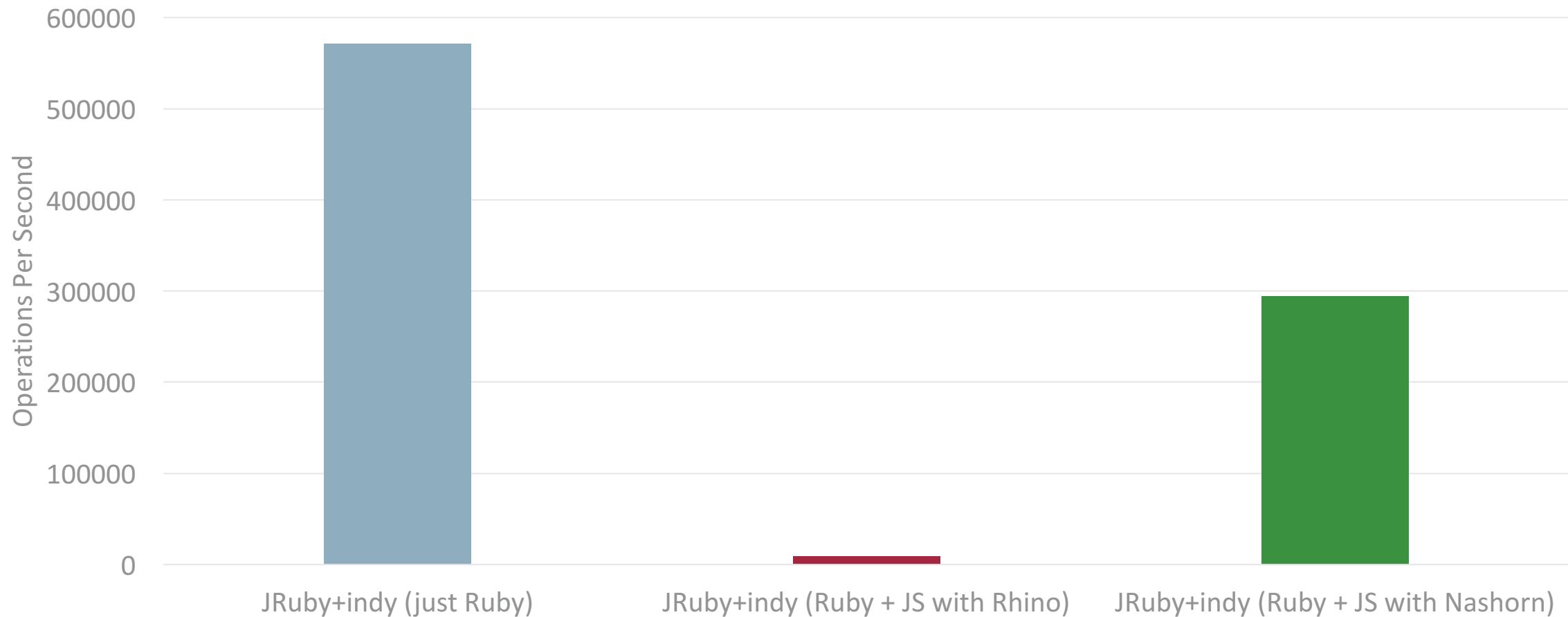


```
factory = javax.script.ScriptEngineManager.new
engine = factory.getEngineByName 'nashorn'
bindings = engine.createBindings

$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
```

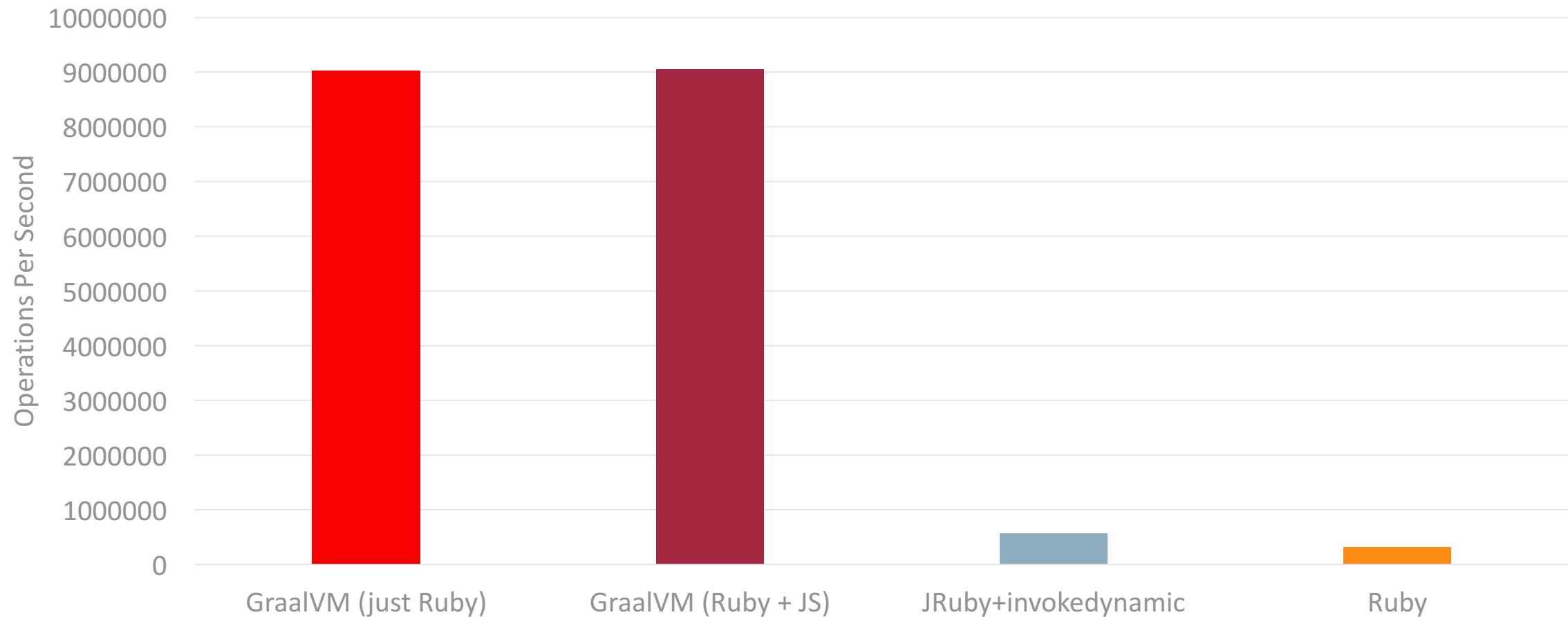
clamp in Ruby and JavaScript with JRuby and Nashorn



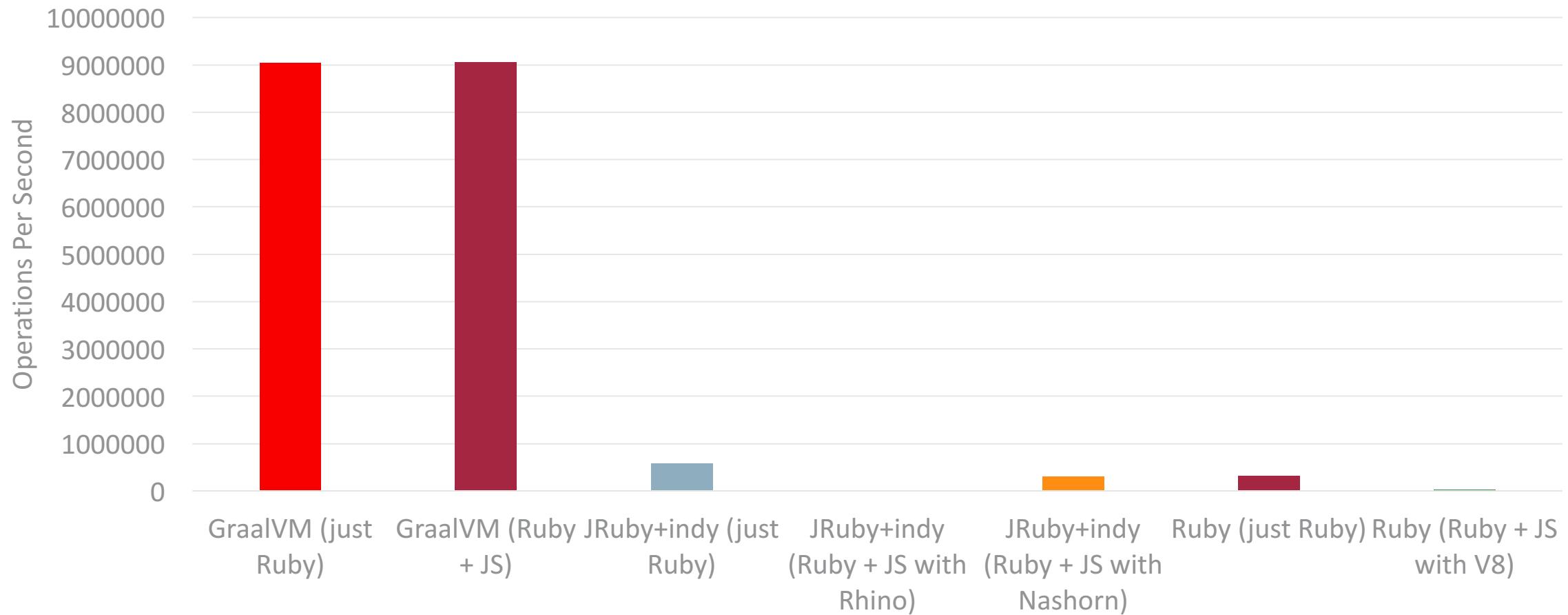
```
function clamp(num, min, max) {
    if (num < min) {
        return min;
    } else if (num > max) {
        return max;
    } else {
        return num;
    }
}

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
```

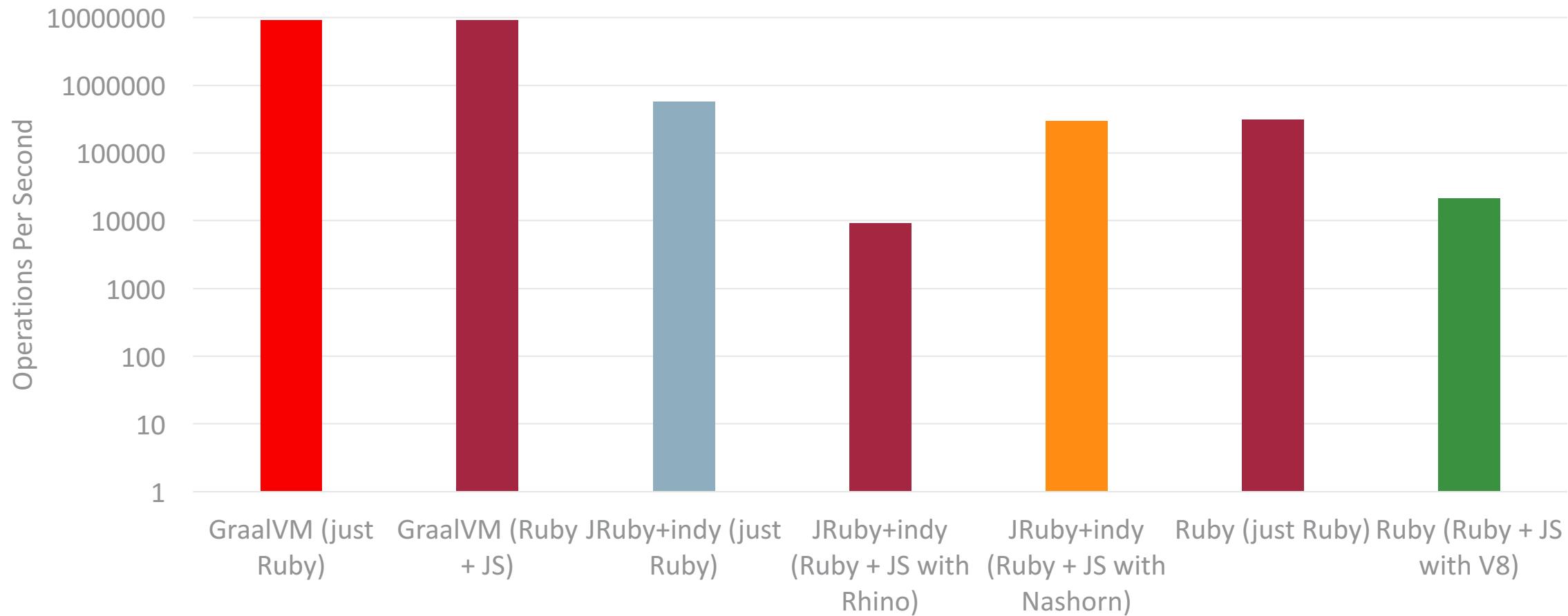
clamp in Ruby and JavaScript with GraalVM



clamp in all configurations



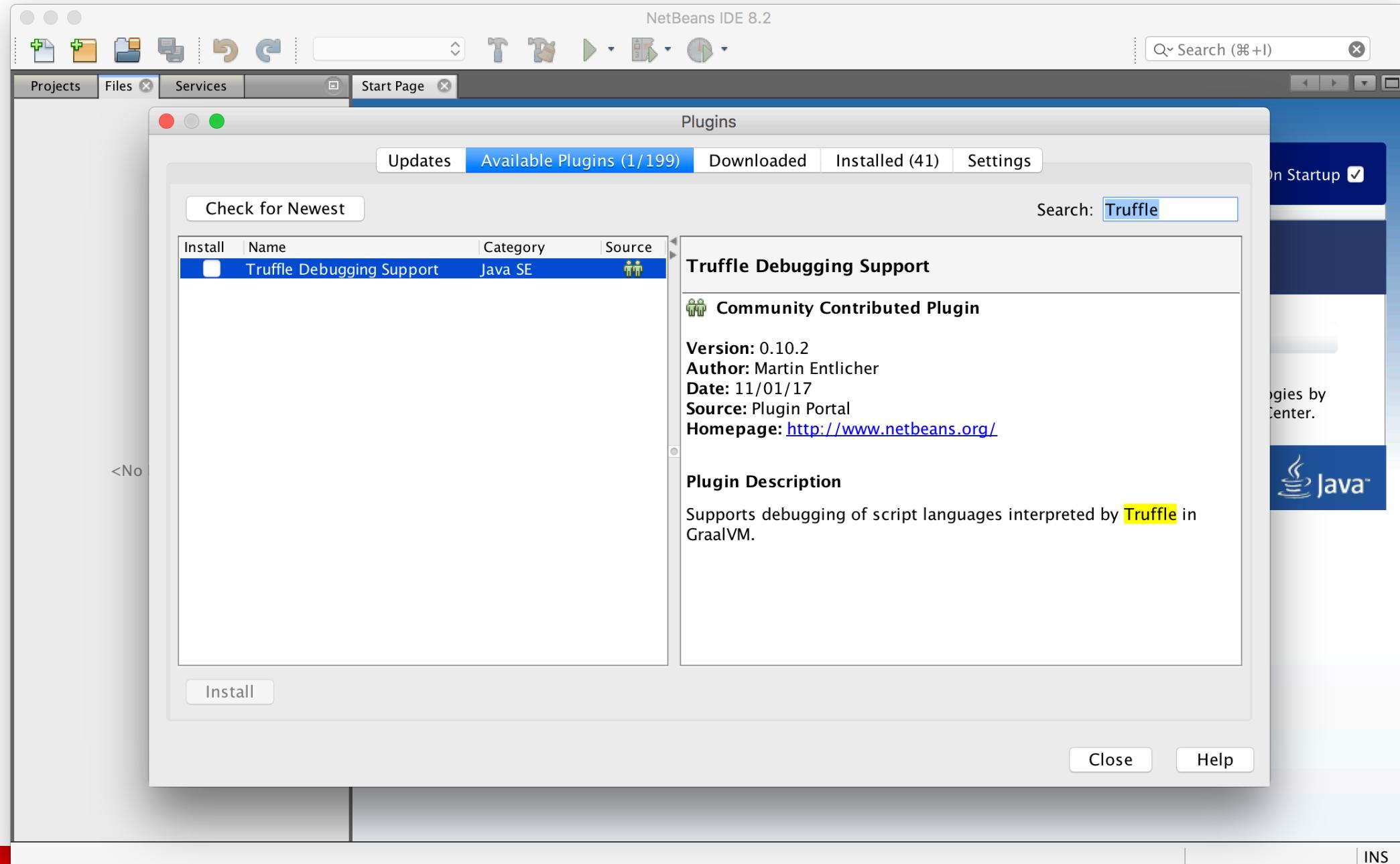
clamp in all configurations



GraalVM and NetBeans



NetBeans 8.2



The screenshot shows a GitHub repository page for 'graalvm / truffleruby'. The page includes a navigation bar with links for 'Pull requests', 'Issues', and 'Gist'. On the right side, there are buttons for 'Unwatch', '43', 'Star', '616', 'Fork', and '22'. Below the navigation bar, there are tabs for 'Code', 'Issues 6', 'Pull requests 3', 'Projects 0', 'Pulse', and 'Graphs'. A search bar on the right contains the text 'Search for 'truffleruby netbeans''. The main content area displays a file named 'netbeans.md' with the following details:

- Branch: master
- Author: chrisseaton
- Commit message: Document how to use NetBeans
- Date: 4bb1b38 17 days ago
- Contributors: 1 contributor
- File statistics: 36 lines (25 sloc) | 1.13 KB
- Actions: Raw, Blame, History, Copy, Edit, Delete

The file content is titled 'NetBeans integration' and contains the following text:

You can debug Ruby programs running in TruffleRuby using the NetBeans IDE.

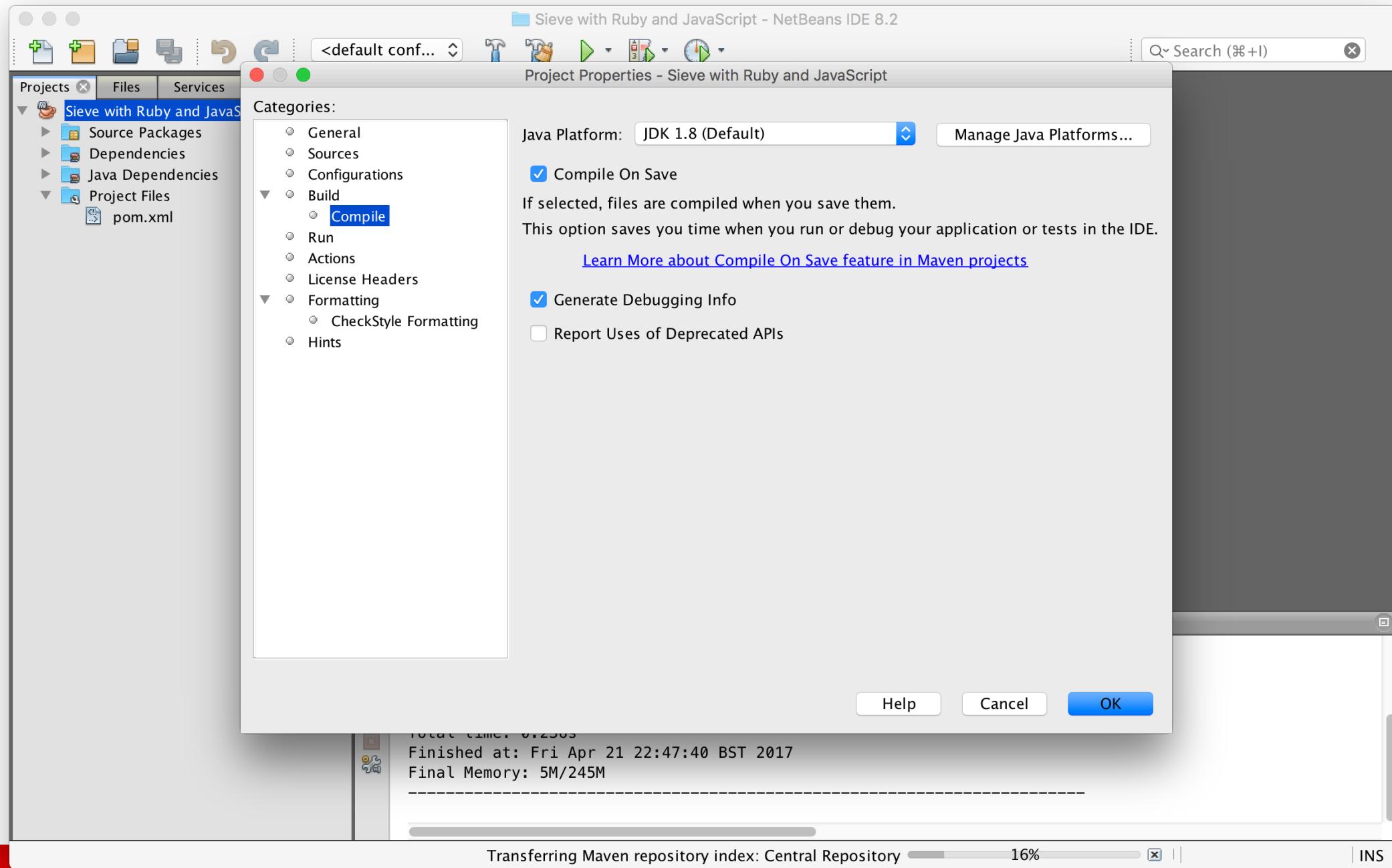
Setup

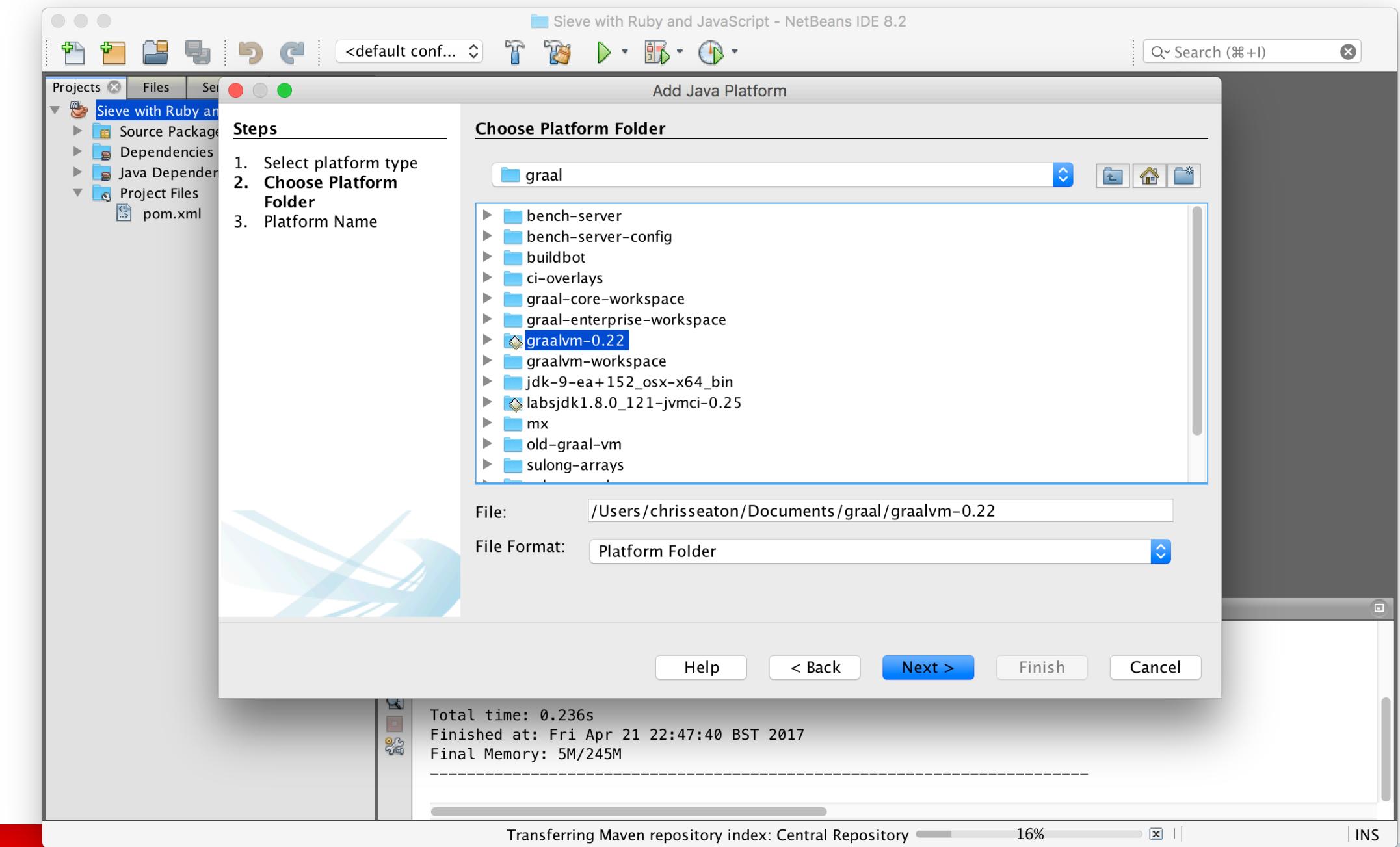
- Install NetBeans 8.2
- Via Tools, Plugins, install 'Truffle Debugging Support'
- [Download GraalVM](#)

To make GraalVM appear as a conventional JVM to directory. run:

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

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





NetBeans IDE 8.2

Projects Files Services <default conf... Main.java sieve.rb sieve.js

Sieve with Ruby and JavaScript

Source Packages

Main.java

```
1 class Natural
2     def initialize
3         @x = 1
4     end
5
6     def next
7         @x += 1
8     end
9 end
10
11 def create
12     Natural.new
13 end
14
15 Truffle::Interop.export("Natural", method(:create))
```

Dependencies

Java Dependencies

Project Files

pom.xml nb-configuration.xml

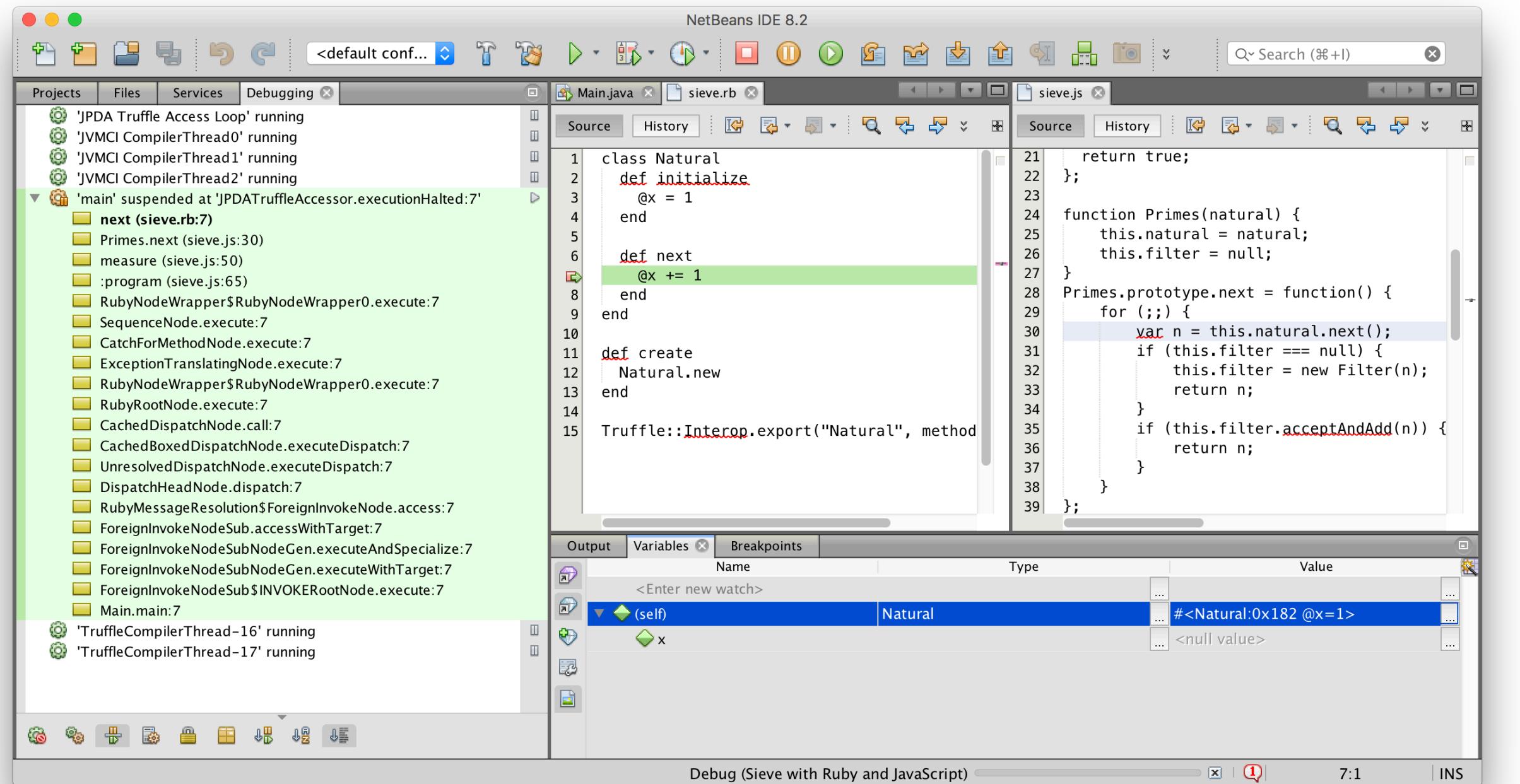
Source History

```
15 }
16     filter = filter.next;
17 }
18 var newFilter = new Filter(n);
19 this.last.next = newFilter;
20 this.last = newFilter;
21 return true;
22 };
23
24 function Primes(natural) {
25     this.natural = natural;
26     this.filter = null;
27 }
28 Primes.prototype.next = function() {
29     for (;;) {
30         var n = this.natural.next();
31         if (this.filter === null) {
32             this.filter = new Filter(n);
33             return n;
34         }
35         if (this.filter.acceptAndAdd(n)) {
36             return n;
37         }
38     }
39 };
40
41 function measure(prntCnt, upto) {
42     var natural = Interop.import('Natural');
43     var primes = new Primes(Interop.execute(natural));
44
45     var log = typeof console !== 'undefined' ? console.
```

Unpacking index for Central Repository 30:37 INS

```
1 class Natural
2     def initialize
3         @x = 1
4     end
5
6     def next
7         @x += 1
8     end
9 end
10
11 def create
12     Natural.new
13 end
14
15 Truffle::Interop.export("Natural", method(:create))
```

```
15     }
16     filter = filter.next;
17 }
18 var newFilter = new Filter(n);
19 this.last.next = newFilter;
20 this.last = newFilter;
21 return true;
22 };
23
24 function Primes(natural) {
25     this.natural = natural;
26     this.filter = null;
27 }
28 Primes.prototype.next = function() {
29     for (;;) {
30         var n = this.natural.next();
31         if (this.filter === null) {
32             this.filter = new Filter(n);
33             return n;
34         }
35         if (this.filter.acceptAndAdd(n)) {
36             return n;
37         }
38     }
39 };
40
41 function measure(prntCnt, upto) {
42     var natural = Interop.import('Natural');
43     var primes = new Primes(Interop.execute(natural));
44 }
```



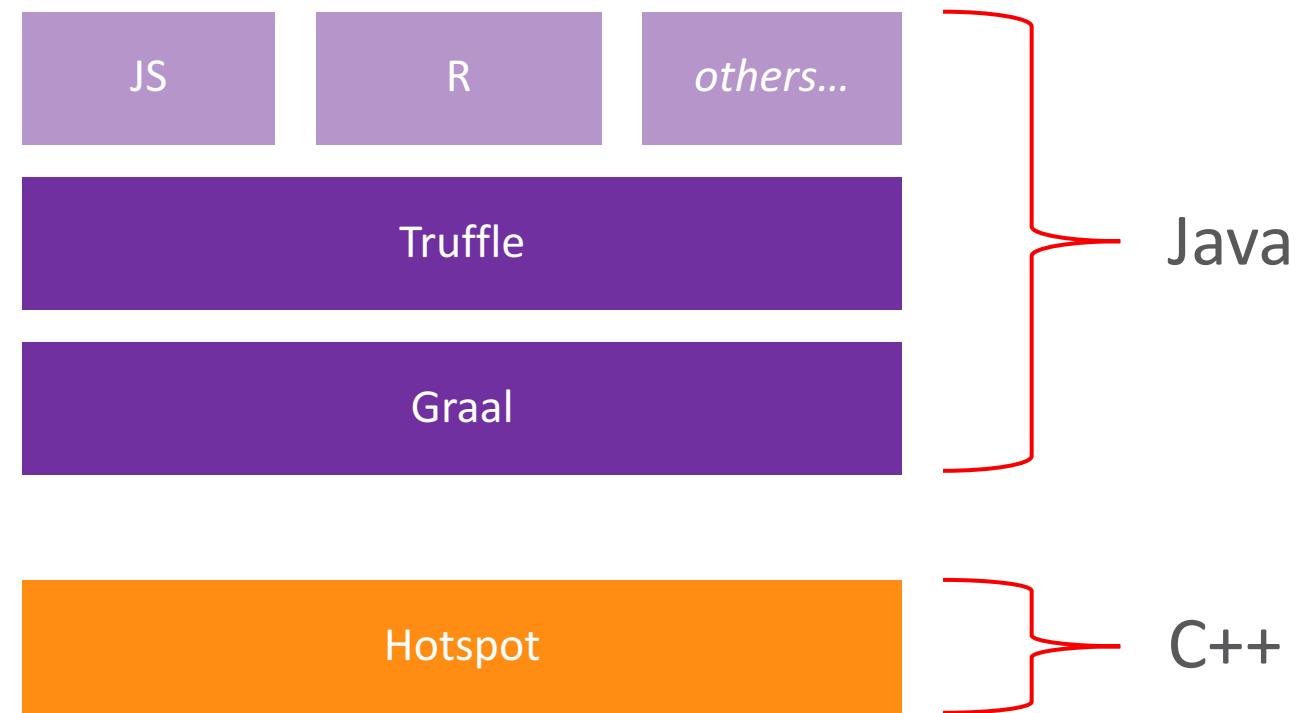
How to use GraalVM

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

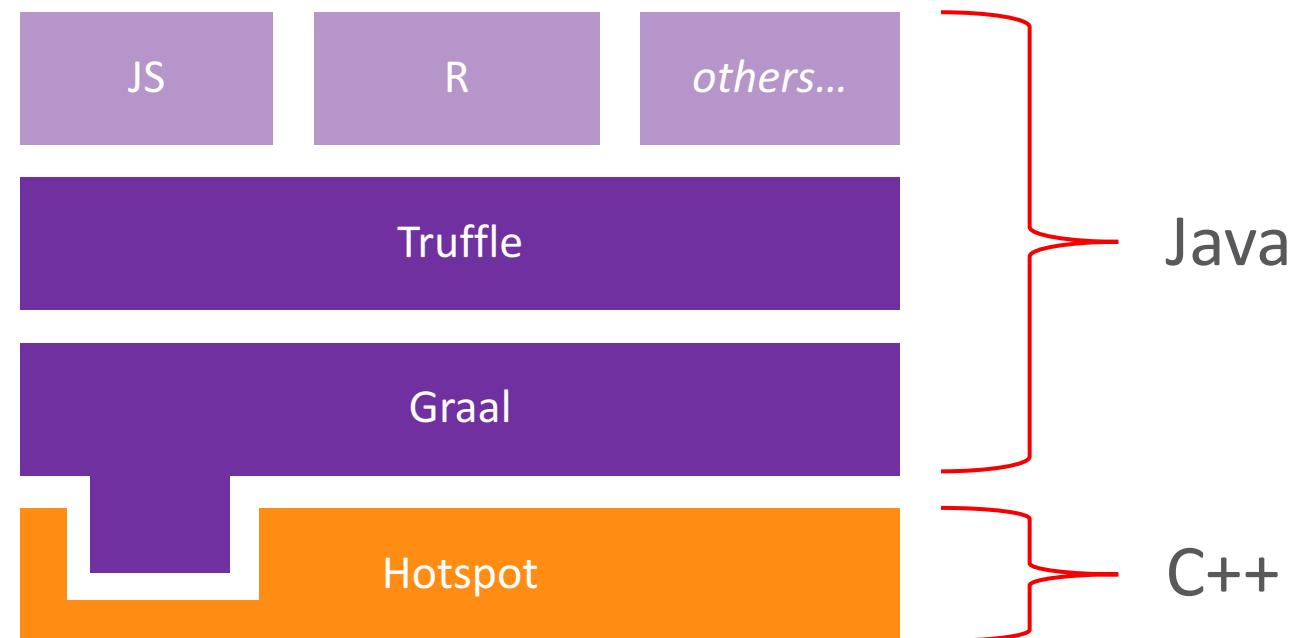


Java 9 – runs on an unmodified JVM

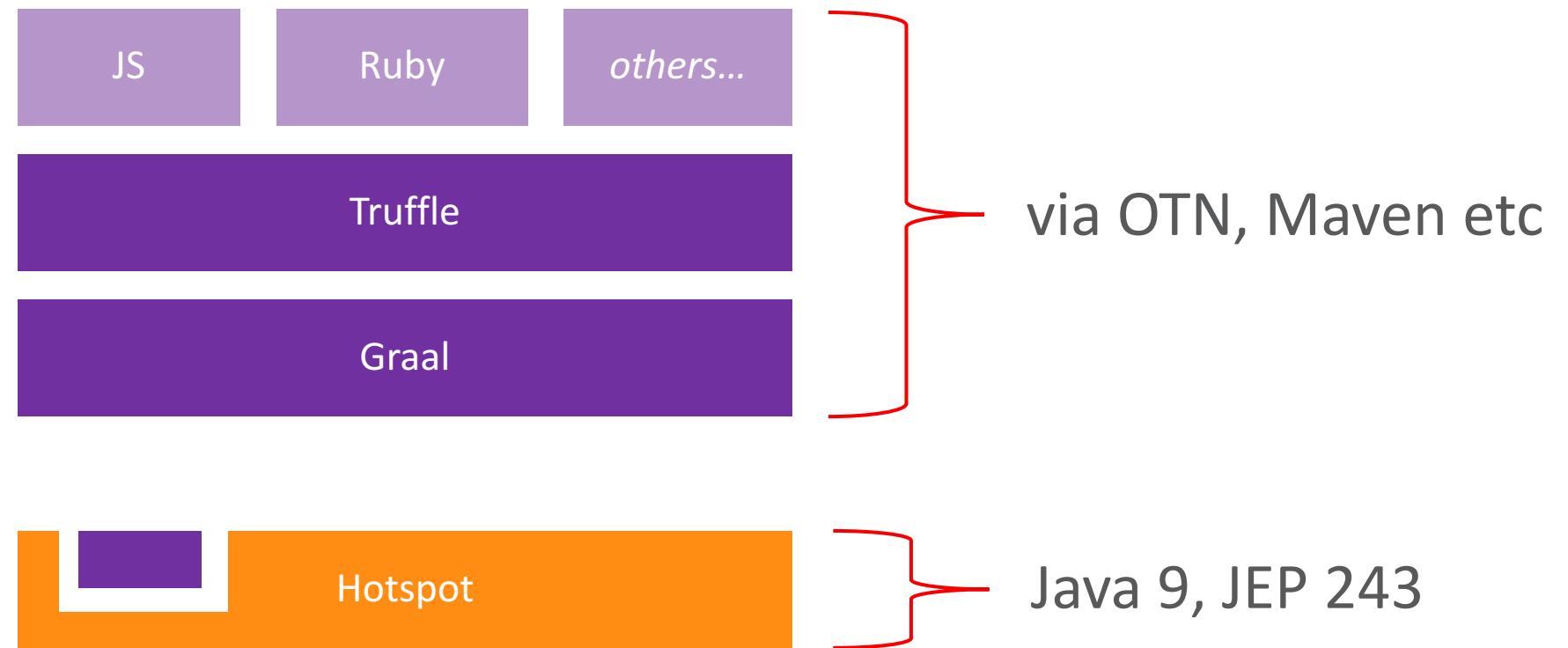


Java 9 – runs on an unmodified JVM

JVMCI
(JVM Compiler Interface)



Java 9 – runs on an unmodified JVM



Where to find more information

Welcome Chris

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
Souffle
Datasets

Overview Java Polyglot **Downloads** Learn More

Oracle Labs GraalVM and JVMCI JDK Downloads

Thank you for downloading this release of the Oracle Labs GraalVM. With this release, one can execute Java applications with Graal, as well as applications written in JavaScript, Ruby, and R, with our Polyglot language engines.

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

[GraalVM preview for Linux \(0.22\)](#)
[GraalVM preview for Linux \(0.22\), with Labs JDK 8](#)
[GraalVM preview for Mac OS X \(0.22\)](#)
[GraalVM preview for Mac OS X \(0.22\), with Labs JDK 8](#)
[GraalVM preview for Solaris SPARC 64-bit \(0.22\)](#)
[GraalVM preview for Solaris SPARC 64-bit \(0.22\), with Labs JDK 8](#)

[labsjdk-8u121-jvmci-0.25-darwin-amd64.tar.gz](#)
[labsjdk-8u121-jvmci-0.25-linux-amd64.tar.gz](#)
[labsjdk-8u121-jvmci-0.25-solaris-sparcv9.tar.gz](#)

How to install GraalVM

Unpack the downloaded *.tar.gz file on your machine. You can then use the java executable to execute Java programs. All those executables are in the bin directory of GraalVM. You might want to add that directory to your operating system's PATH.

More detailed getting started instructions are available in the README files in the download. The README files for the language engines can be found in jre/language.

About this OTN Release

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

 This organization Search

Pull requests Issues Gist

GitHub, Inc.

Search for 'github graalvm'

<https://graalvm.github.io>

Graal Multi-Language VM
Next generation compilation technology supporting Java, Ruby, R, JavaScript, LLVM, and more.

Repositories People 38 Teams 2

Filters Find a repository...

sulong Java ★ 211 ⚡ 19
Sulong, a dynamic runtime for LLVM-based languages.
Updated 6 minutes ago

graal-core Java ★ 122 ⚡ 33
Graal Compiler & Truffle Partial evaluator.
Updated 31 minutes ago

mx Python ★ 13 ⚡ 26

People 38 >



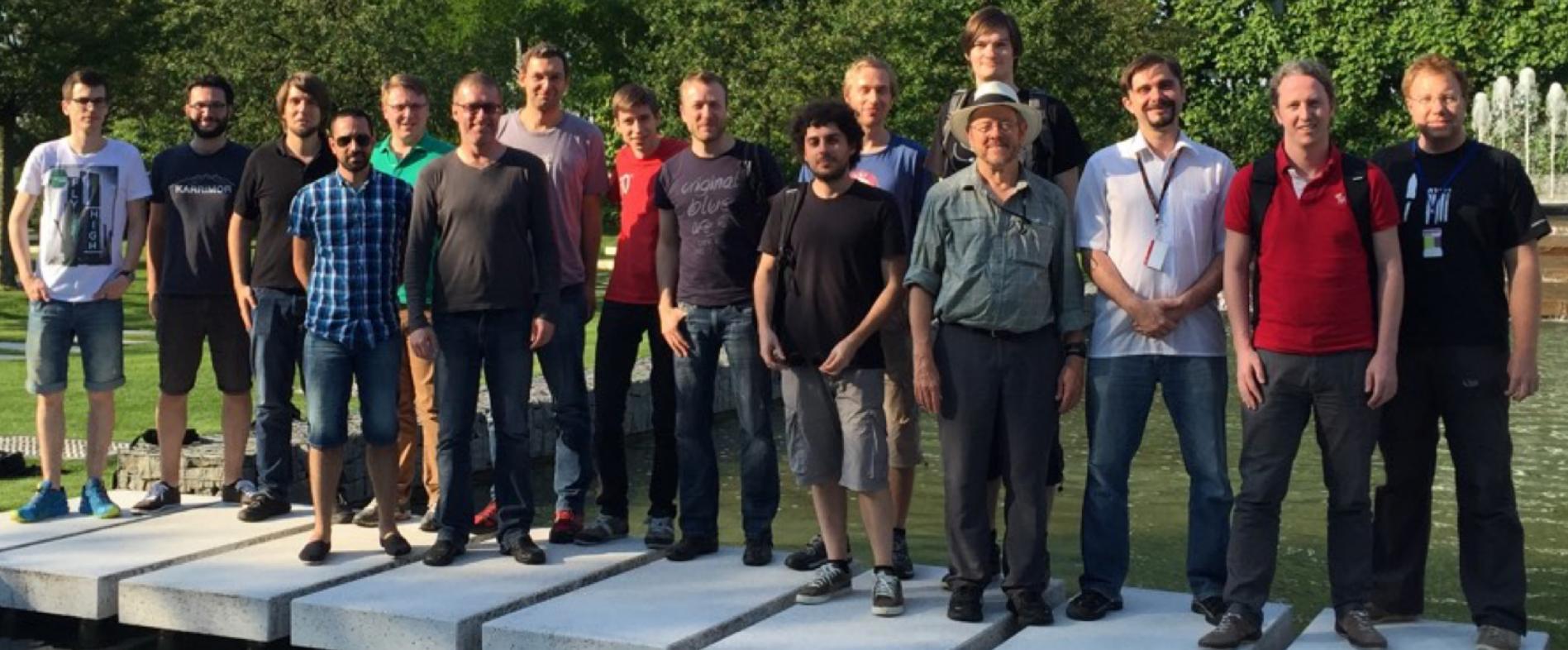
github.com/graalvm

@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 Hummer
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 Daloz
Josef Eisl
Thomas Feichtinger
Matthias Grimmer
Christian Häubl
Josef Haider
Christian Huber
Stefan Marr
Manuel Rigger
Stefan Rumzucker
Bernhard Urban

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 Edinburgh

Christophe Dubach
Juan José Fumero Alfonso
Ranjeet Singh
Toomas Remmelg

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®