GraalVM™

Run Programs Faster Anywhere

Oleg Šelajev
Developer Advocate, GraalVM team, Oracle Labs
@shelajev
GOTO Berlin 2018

ORACLE®

Please

Ask questions through the app

Rate Session

Thank you!

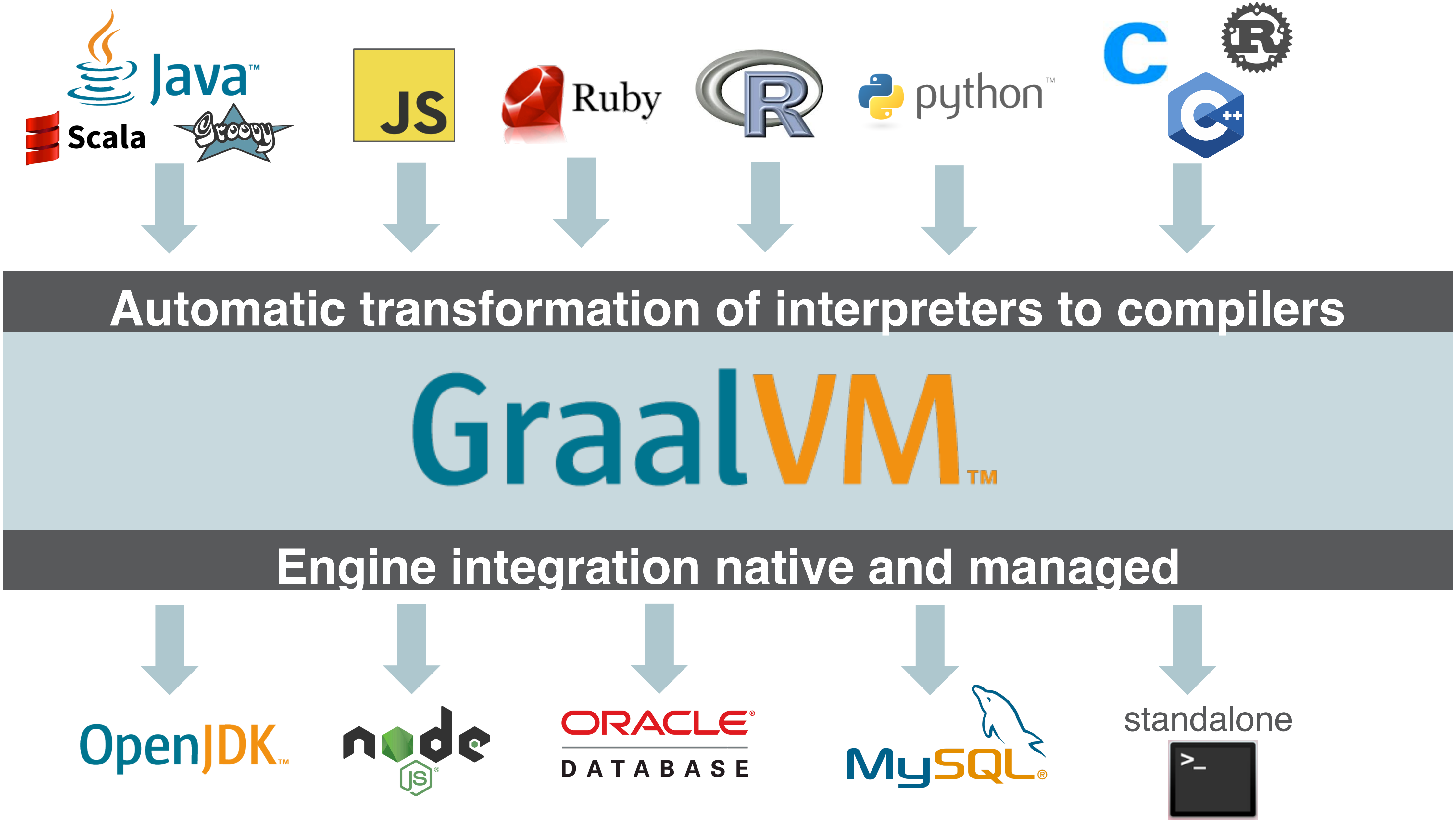
@GOTOber

gotober.com

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. 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.

GraalVM™



Automatic transformation of interpreters to compilers

GraalVM™

Engine integration native and managed

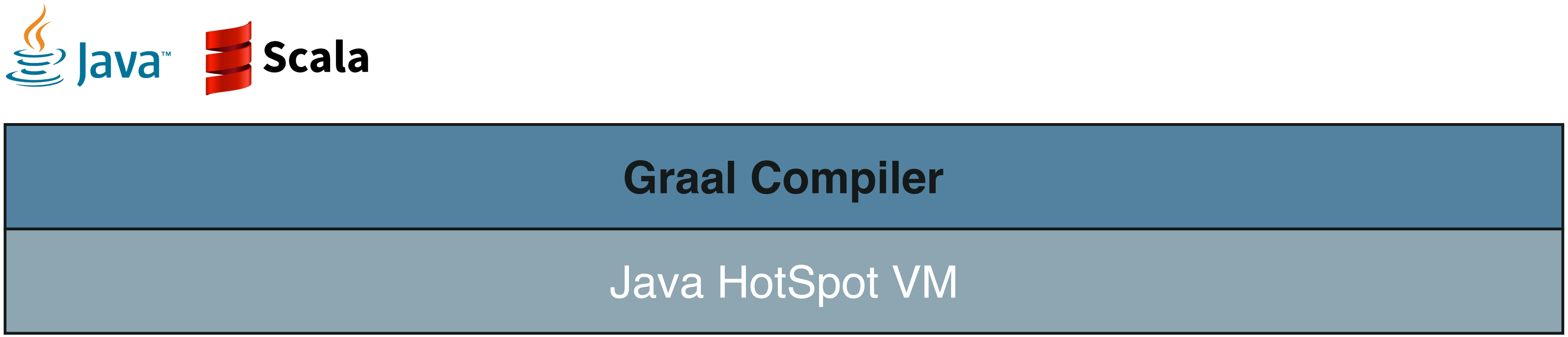
ORACLE®

Top 10 Things To Do With GraalVM

1. High-performance modern Java
2. Low-footprint, fast-startup Java
3. Combine JavaScript, Java, Ruby, and R
4. Run native languages on the JVM
5. Tools that work across all languages
6. Extend a JVM-based application
7. Extend a native application
8. Java code as a native library
9. Polyglot in the database
10. Create your own language

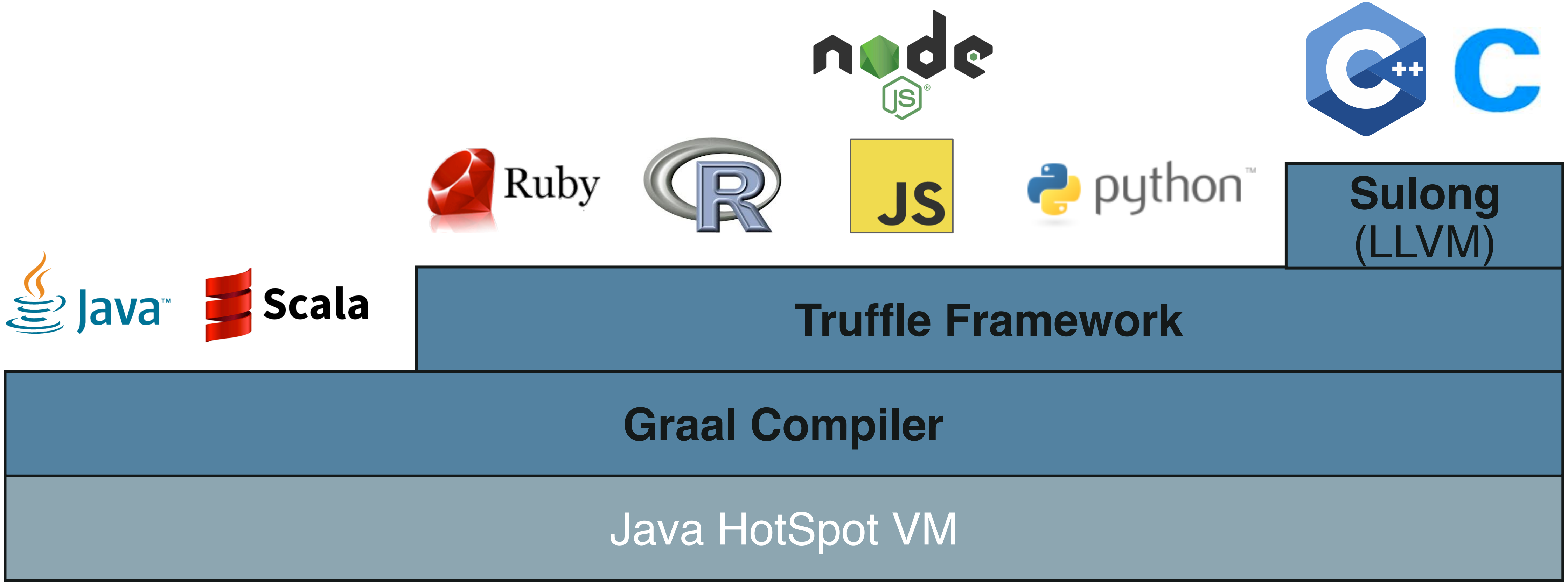
medium.com/graalvm/graalvm-ten-things-12d9111f307d

ORACLE



Graal Compiler

Java HotSpot VM



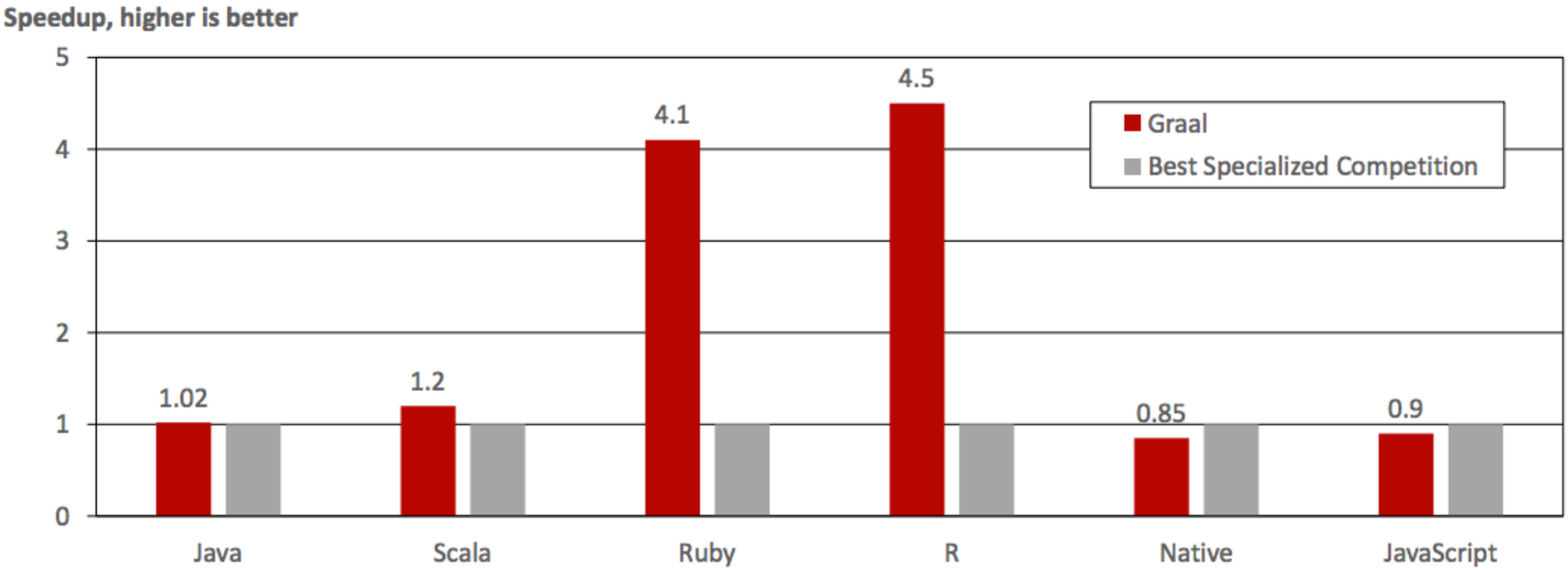
Sulong (LLVM)

Truffle Framework

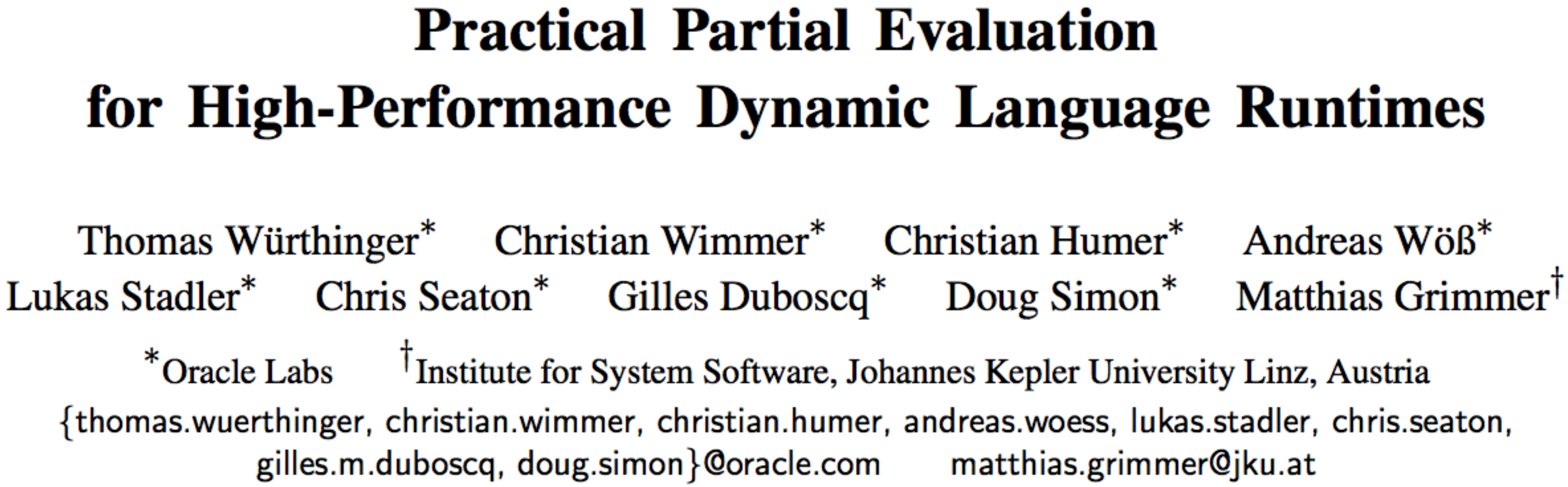
Graal Compiler

Java HotSpot VM

Performance: Graal VM



Performance relative to:
HotSpot/Server, HotSpot/Server running JRuby, GNU R, LLVM AOT compiled, V8



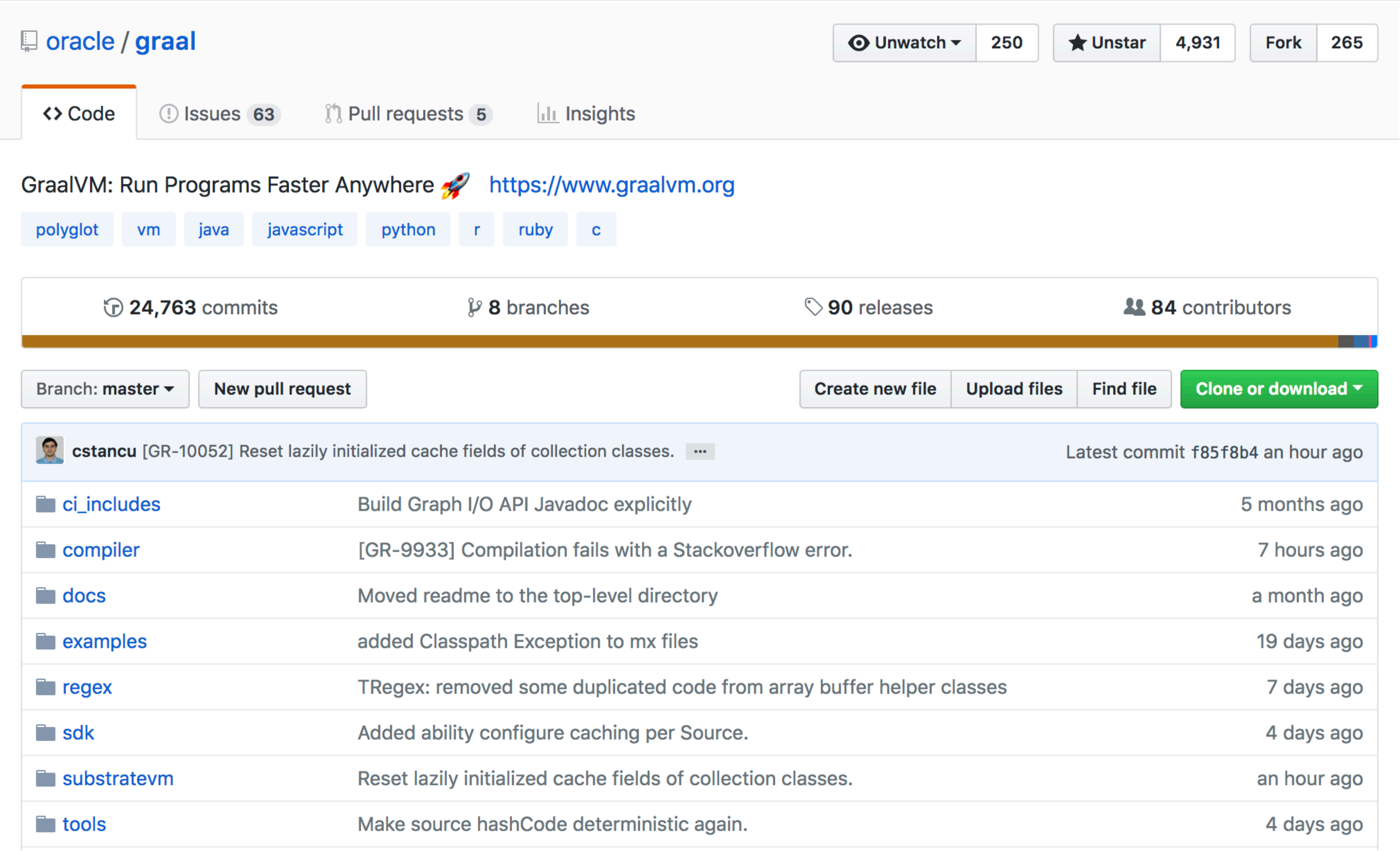
Practical Partial Evaluation for High-Performance Dynamic Language Runtimes

Thomas Würthinger[*] Christian Wimmer[*] Christian Humer[*] Andreas Wöß[*]
Lukas Stadler[*] Chris Seaton[*] Gilles Duboscq[*] Doug Simon[*] Matthias Grimmer[†]

[*]Oracle Labs [†]Institute for System Software, Johannes Kepler University Linz, Austria

{thomas.wuerthinger, christian.wimmer, christian.humer, andreas.woess, lukas.stadler, chris.seaton,
gilles.m.duboscq, doug.simon}@oracle.com matthias.grimmer@jku.at

http://chrisseaton.com/rubytruffle/pldi17-truffle/pldi17-truffle.pdf



oracle / graal

GraalVM: Run Programs Faster Anywhere 🚀 https://www.graalvm.org

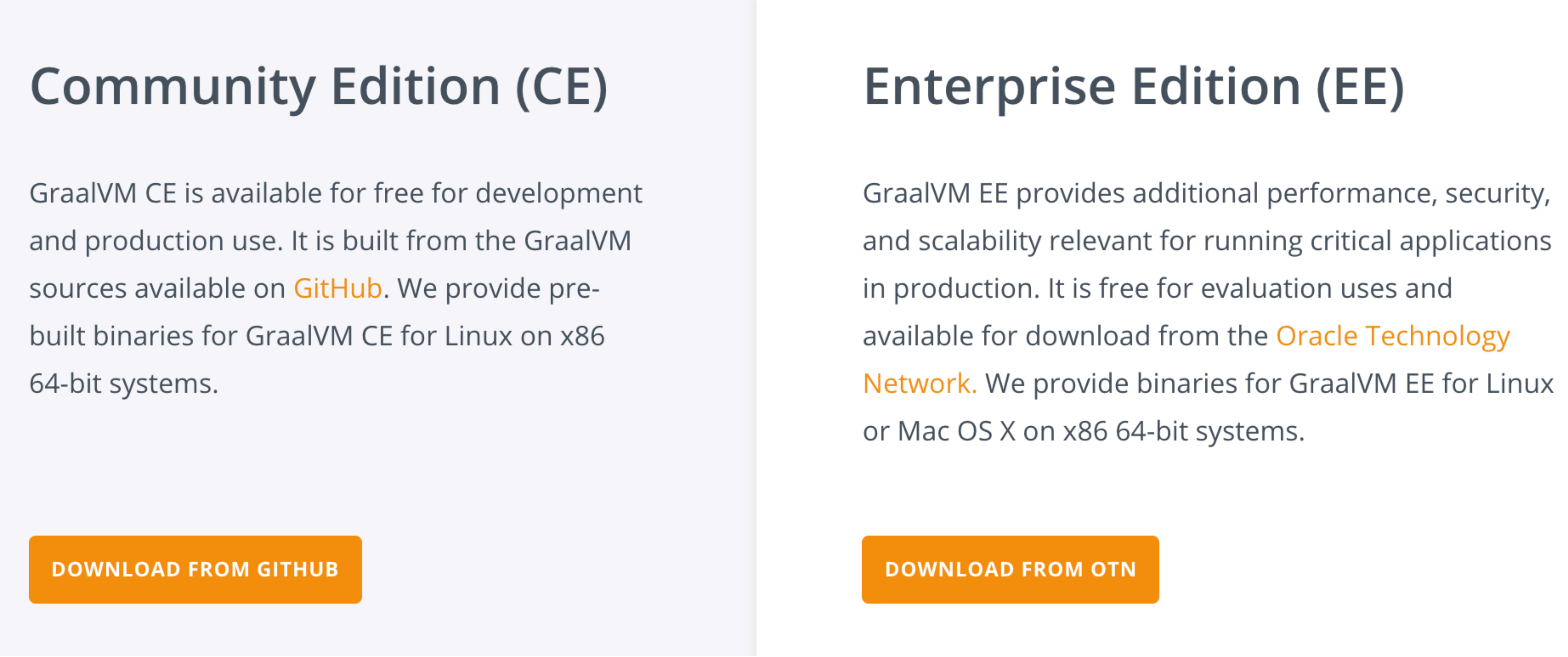
polyglot vm java javascript python r ruby c

24,763 commits | 8 branches | 90 releases | 84 contributors

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

cstancu [GR-10052] Reset lazily initialized cache fields of collection classes. … Latest commit f85f8b4 an hour ago

ci_includes	Build Graph I/O API Javadoc explicitly	5 months ago
compiler	[GR-9933] Compilation fails with a Stackoverflow error.	7 hours ago
docs	Moved readme to the top-level directory	a month ago
examples	added Classpath Exception to mx files	19 days ago
regex	TRegex: removed some duplicated code from array buffer helper classes	7 days ago
sdk	Added ability configure caching per Source.	4 days ago
substratevm	Reset lazily initialized cache fields of collection classes.	an hour ago
tools	Make source hashCode deterministic again.	4 days ago



Community Edition (CE)

GraalVM CE is available for free for development and production use. It is built from the GraalVM sources available on GitHub. We provide pre-built binaries for GraalVM CE for Linux on x86 64-bit systems.

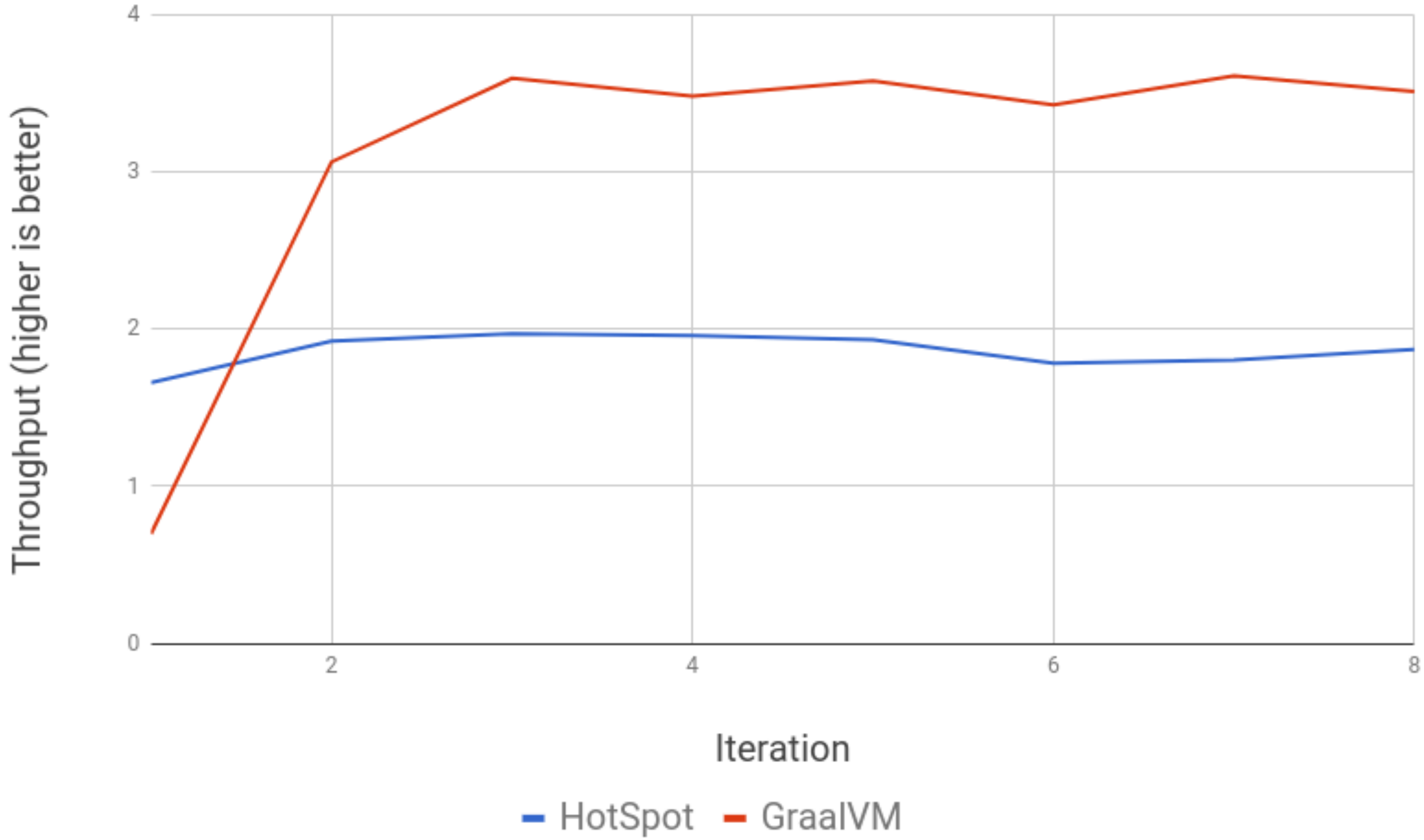
Enterprise Edition (EE)

GraalVM EE provides additional performance, security, and scalability relevant for running critical applications in production. It is free for evaluation uses and available for download from the Oracle Technology Network. We provide binaries for GraalVM EE for Linux or Mac OS X on x86 64-bit systems.

ORACLE

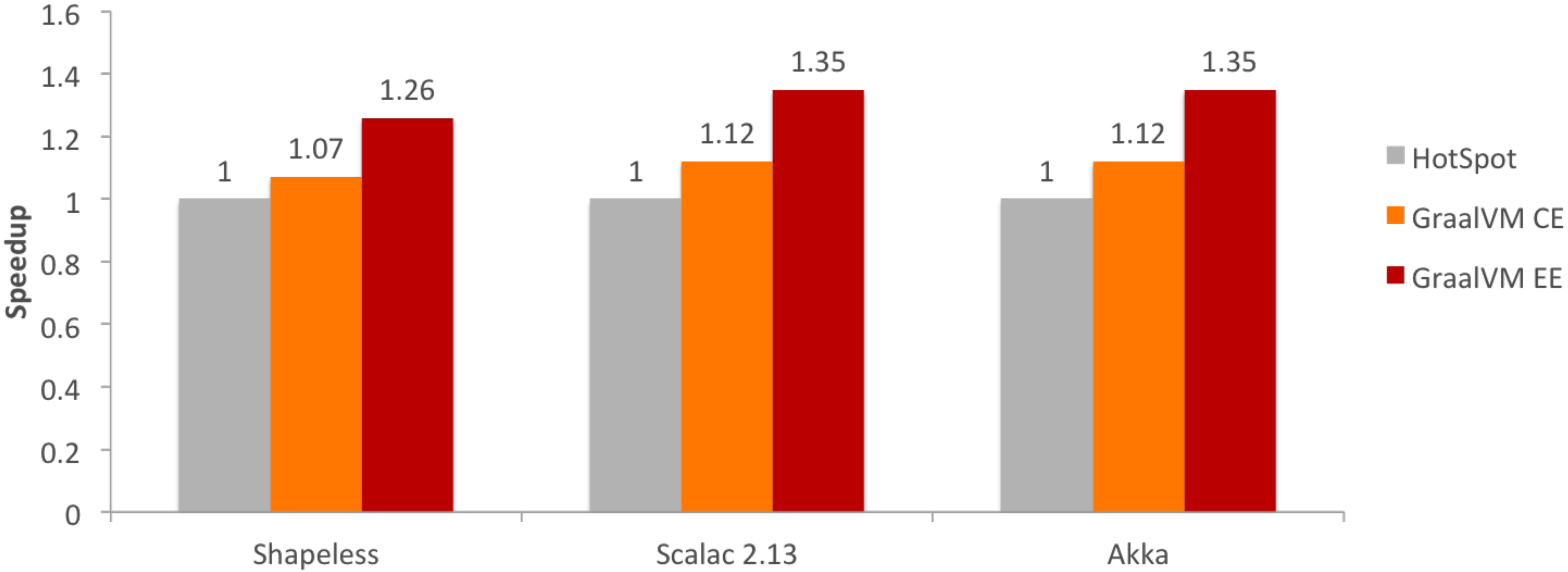
Jenga

JavaScrabble.run(); (1.9x)



https://medium.com/graalvm/stream-api-performance-with-graalvm-be6cfe7fbb52

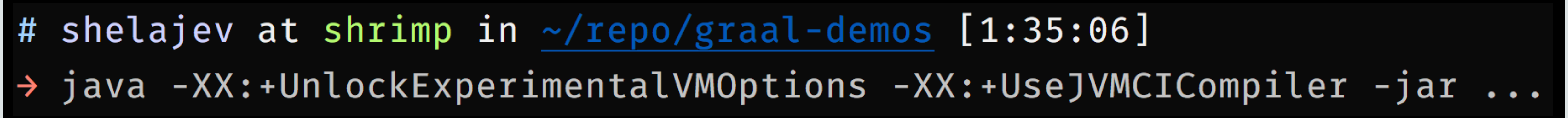
sbt > clean; compile;



https://medium.com/graalvm/compiling-scala-faster-with-graalvm-86c5c0857fa3

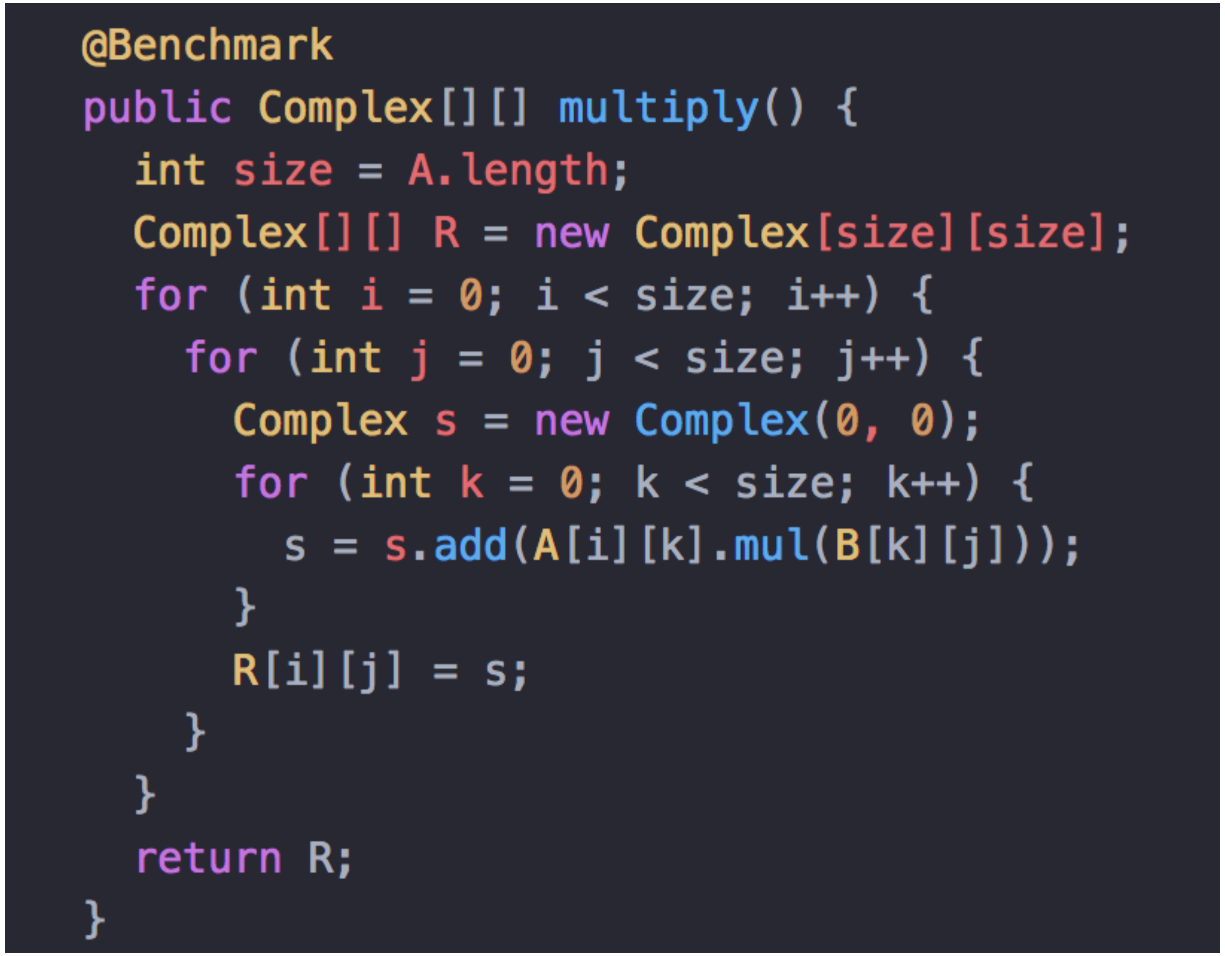
OpenJDK

GraalVM™



```
# shelajev at shrimp in ~/repo/graal-demos [1:35:06]
→ java -XX:+UnlockExperimentalVMOptions -XX:+UseJVMCICompiler -jar ...
```

Matrix multiplication



```
@Benchmark
public Complex[][] multiply() {
  int size = A.length;
  Complex[][] R = new Complex[size][size];
  for (int i = 0; i < size; i++) {
    for (int j = 0; j < size; j++) {
      Complex s = new Complex(0, 0);
      for (int k = 0; k < size; k++) {
        s = s.add(A[i][k].mul(B[k][j]));
      }
      R[i][j] = s;
    }
  }
  return R;
}
```

https://www.youtube.com/watch?v=RFF2SfPMfpk

Matrix multiplication

valhallaBench.Multiply.multiply (lower is better)

JDK 11	(size)	Mode	Cnt	Score		Error	Units
valhallaBench.Multiply.multiply	100	avgt	3	7944.935	±	1963.931	us/op
JDK11 + Graal	(size)	Mode	Cnt	Score		Error	Units
valhallaBench.Multiply.multiply	100	avgt	3	3450.944	±	1130.123	us/op
GraalVM EE 1.0-rc8	(size)	Mode	Cnt	Score		Error	Units
valhallaBench.Multiply.multiply	100	avgt	3	3134.066	±	518812	us/op

Matrix multiplication

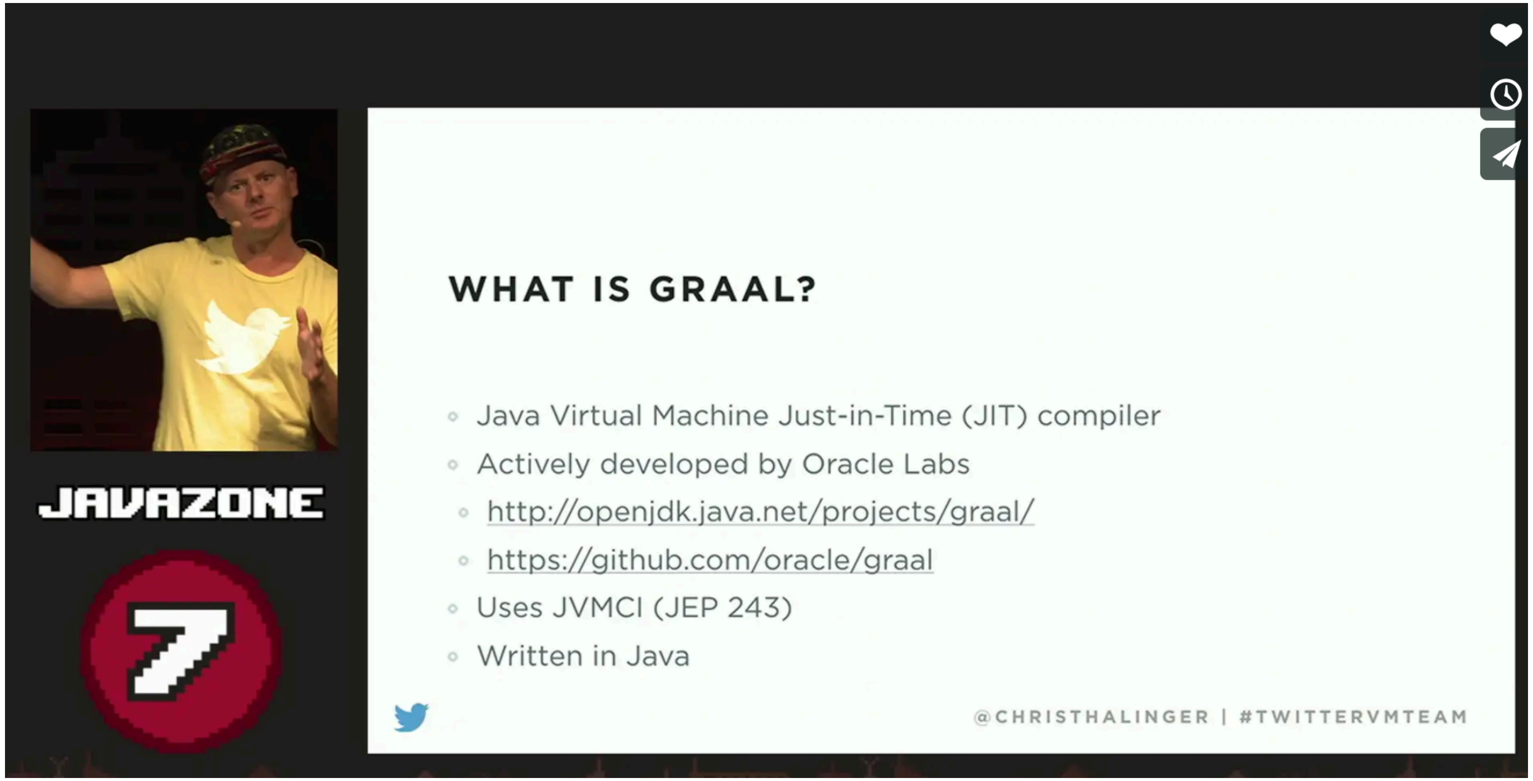
valhallaBench.Multiply.multiply (lower is better)

JDK 11	(size)	Mode	Cnt	Score		Error	Units
valhallaBench.Multiply.multiply	100	avgt	3	7944.935	±	1963.931	us/op
JDK11 + Graal	(size)	Mode	Cnt	Score		Error	Units
valhallaBench.Multiply.multiply	100	avgt	3	3450.944	±	1130.123	us/op
				2.3x			
GraalVM EE 1.0-rc8	(size)	Mode	Cnt	Score		Error	Units
valhallaBench.Multiply.multiply	100	avgt	3	3134.066	±	518812	us/op
				2.5x			

Graal: How to use the new JVM JIT compiler in real life

Chris Thalinger

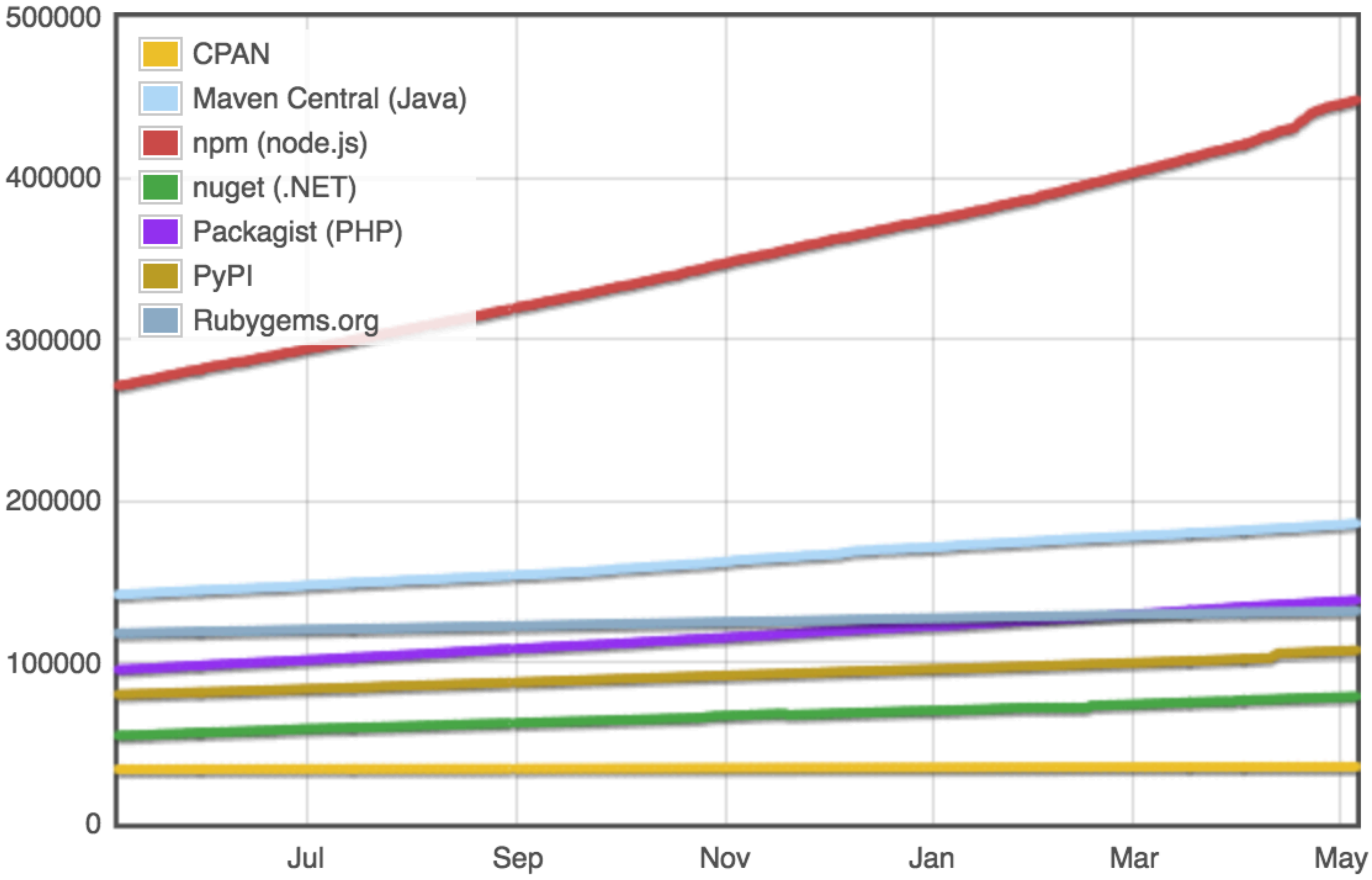
@christhalinger



WHAT IS GRAAL?

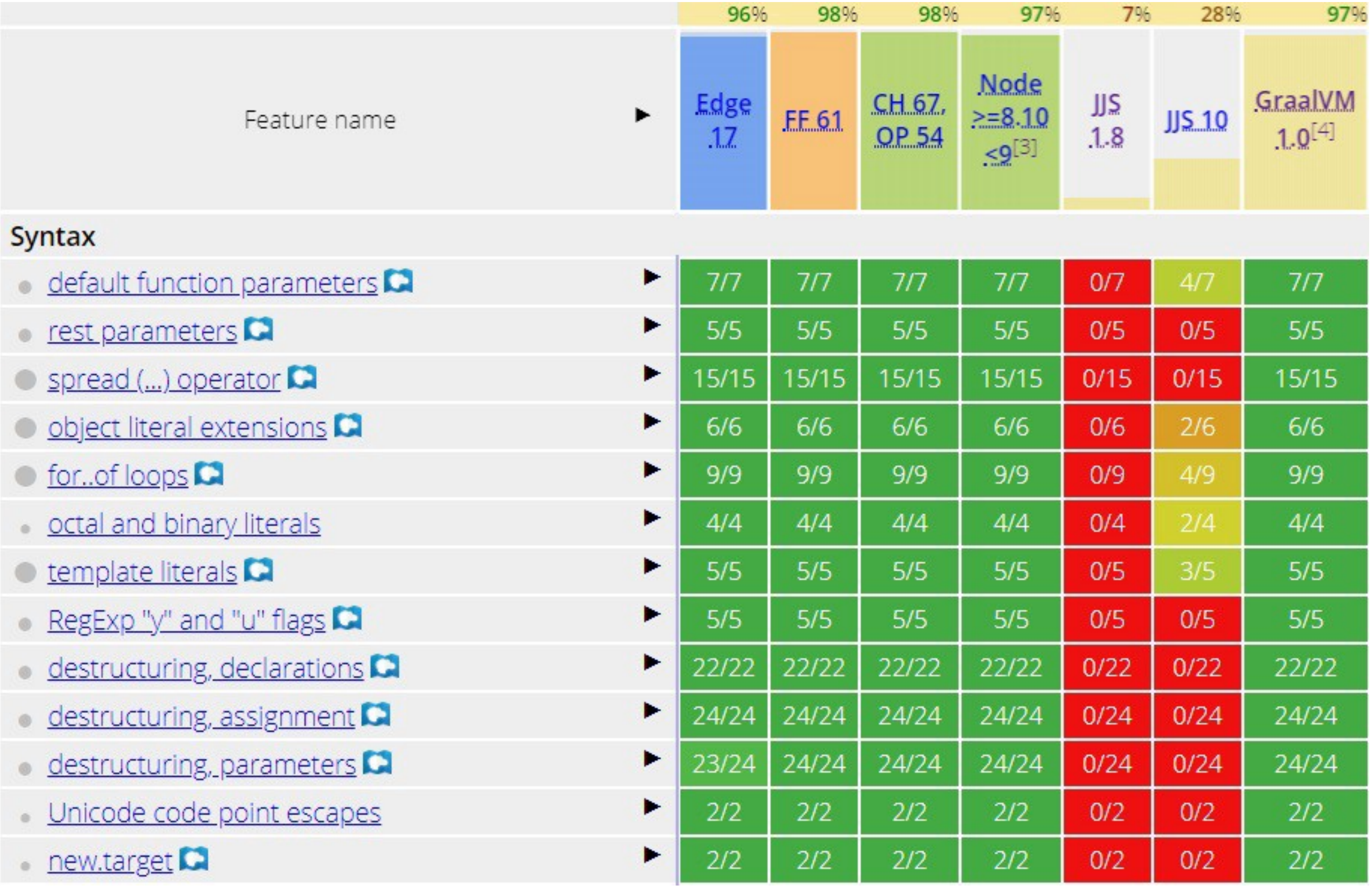
- Java Virtual Machine Just-in-Time (JIT) compiler
- Actively developed by Oracle Labs
  - http://openjdk.java.net/projects/graal/
  - https://github.com/oracle/graal
- Uses JVMCI (JEP 243)
- Written in Java

@CHRISTHALINGER | #TWITTERVMTEAM

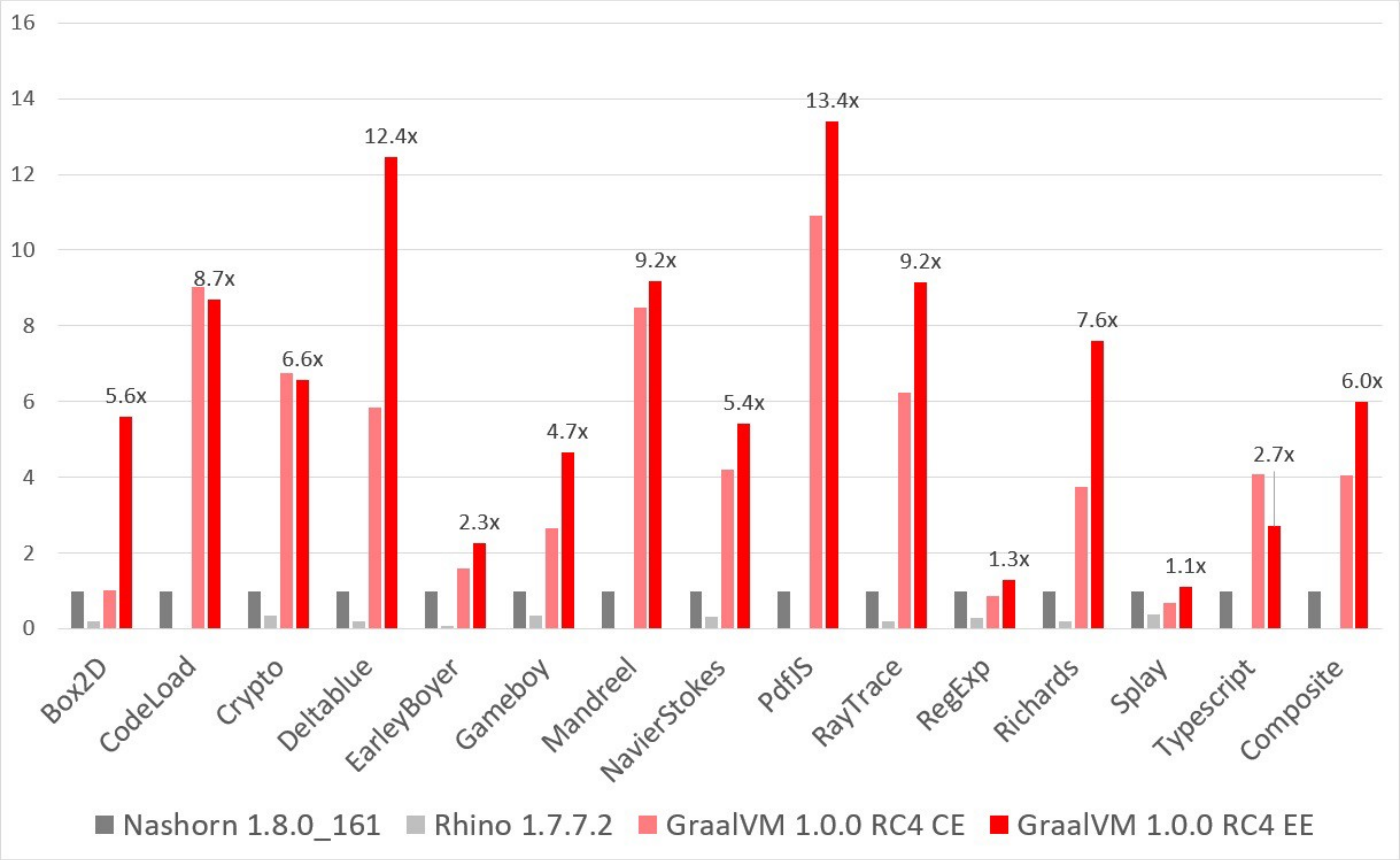

500000
400000
300000
200000
100000
0
Jul
Sep
Nov
Jan
Mar
May
CPAN
Maven Central (Java)
npm (node.js)
nuget (.NET)
Packagist (PHP)
PyPI
Rubygems.org

Jenga
Jenga
Jenga
Jenga
Jenga
Jenga

ECMAScript 6 compatibility



	96%	98%	98%	97%	7%	28%	97%
Feature name ▶	Edge 17	FF 61	CH 67, OP 54	Node >=8.10 <9[3]	JJS 1.8	JJS 10	GraalVM 1.0[4]
Syntax							
default function parameters ▶	7/7	7/7	7/7	7/7	0/7	4/7	7/7
rest parameters ▶	5/5	5/5	5/5	5/5	0/5	0/5	5/5
spread (...) operator ▶	15/15	15/15	15/15	15/15	0/15	0/15	15/15
object literal extensions ▶	6/6	6/6	6/6	6/6	0/6	2/6	6/6
for..of loops ▶	9/9	9/9	9/9	9/9	0/9	4/9	9/9
octal and binary literals ▶	4/4	4/4	4/4	4/4	0/4	2/4	4/4
template literals ▶	5/5	5/5	5/5	5/5	0/5	3/5	5/5
RegExp "y" and "u" flags ▶	5/5	5/5	5/5	5/5	0/5	0/5	5/5
destructuring, declarations ▶	22/22	22/22	22/22	22/22	0/22	0/22	22/22
destructuring, assignment ▶	24/24	24/24	24/24	24/24	0/24	0/24	24/24
destructuring, parameters ▶	23/24	24/24	24/24	24/24	0/24	0/24	24/24
Unicode code point escapes ▶	2/2	2/2	2/2	2/2	0/2	0/2	2/2
new.target ▶	2/2	2/2	2/2	2/2	0/2	0/2	2/2


16
14
12
10
8
6
4
2
0
5.6x
8.7x
6.6x
12.4x
2.3x
4.7x
9.2x
5.4x
13.4x
9.2x
1.3x
7.6x
1.1x
2.7x
6.0x
Box2D
CodeLoad
Crypto
Deltablue
EarleyBoyer
Gameboy
Mandreel
NavierStokes
PdfJS
RayTrace
RegExp
Richards
Splay
Typescript
Composite
Nashorn 1.8.0_161
Rhino 1.7.7.2
GraalVM 1.0.0 RC4 CE
GraalVM 1.0.0 RC4 EE

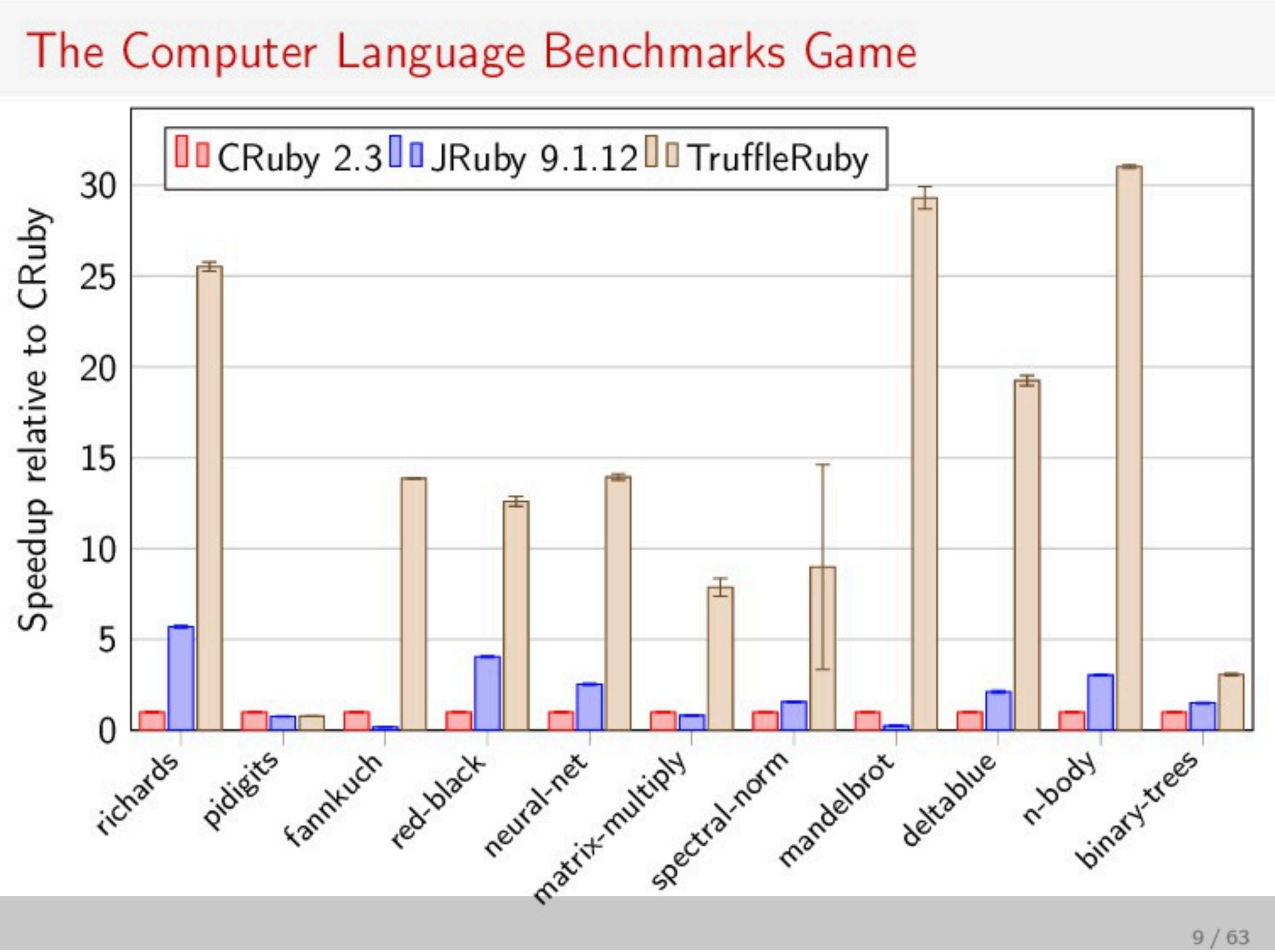
Graal.js on JDK11

```xml
<dependency>
    <groupId>org.graalvm.sdk</groupId>
    <artifactId>graal-sdk</artifactId>
    <version>${graalvm.version}</version>
</dependency>
<dependency>
    <groupId>org.graalvm.truffle</groupId>
    <artifactId>truffle-api</artifactId>
    <version>${graalvm.version}</version>
    <scope>runtime</scope>
</dependency>
<dependency>
    <groupId>org.graalvm.js</groupId>
    <artifactId>js</artifactId>
    <version>${graalvm.version}</version>
    <scope>runtime</scope>
</dependency>
```

```xml
<dependency>
    <groupId>org.graalvm.js</groupId>
    <artifactId>js-scriptengine</artifactId>
    <version>${graalvm.version}</version>
    <scope>runtime</scope>
</dependency>
<dependency>
    <groupId>org.graalvm.tools</groupId>
    <artifactId>profiler</artifactId>
    <version>${graalvm.version}</version>
    <scope>runtime</scope>
</dependency>
<dependency>
    <groupId>org.graalvm.tools</groupId>
    <artifactId>chromeinspector</artifactId>
    <version>${graalvm.version}</version>
    <scope>runtime</scope>
</dependency>
```

https://github.com/graalvm/graal-js-jdk11-maven-demo

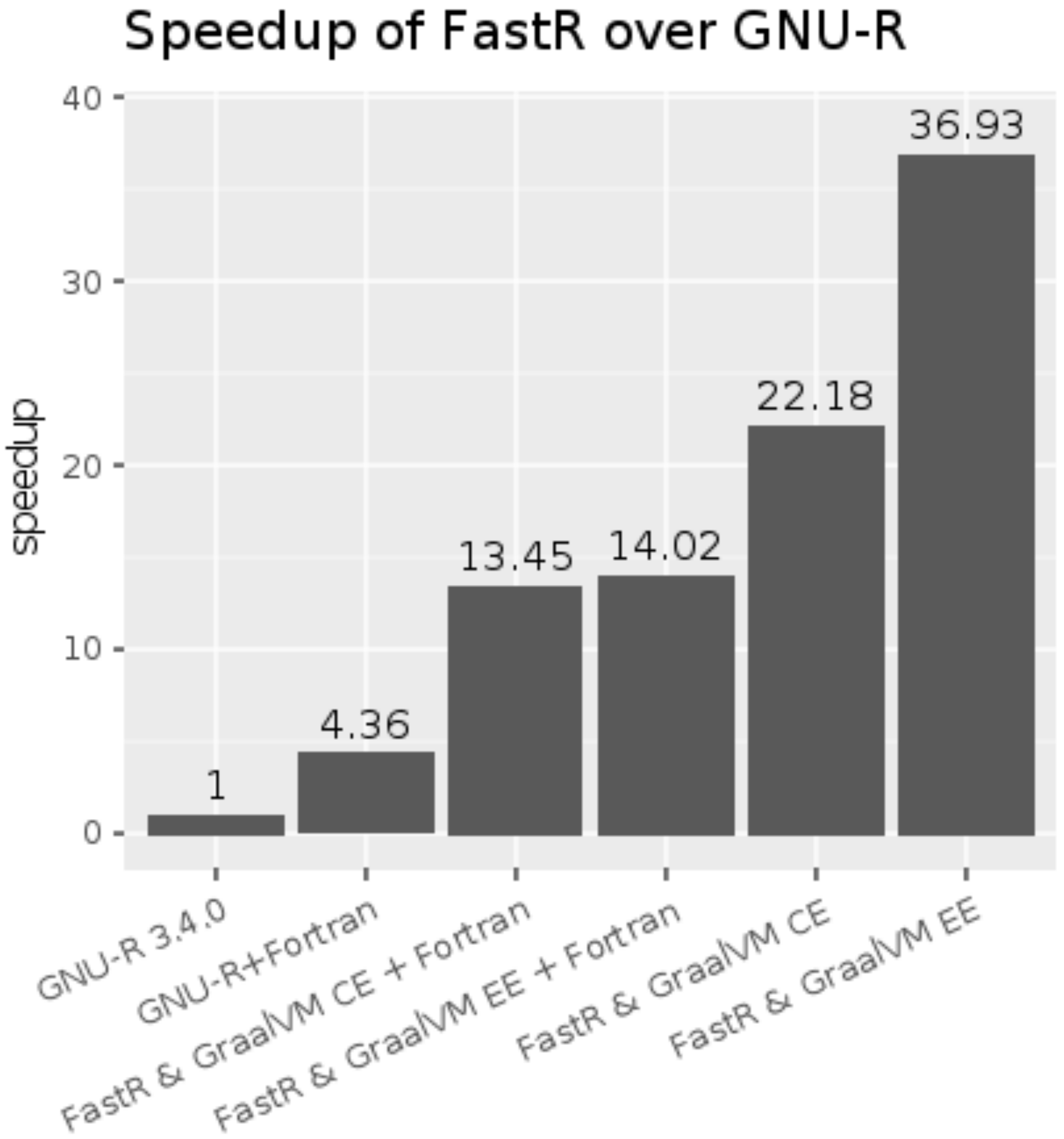
ORACLE



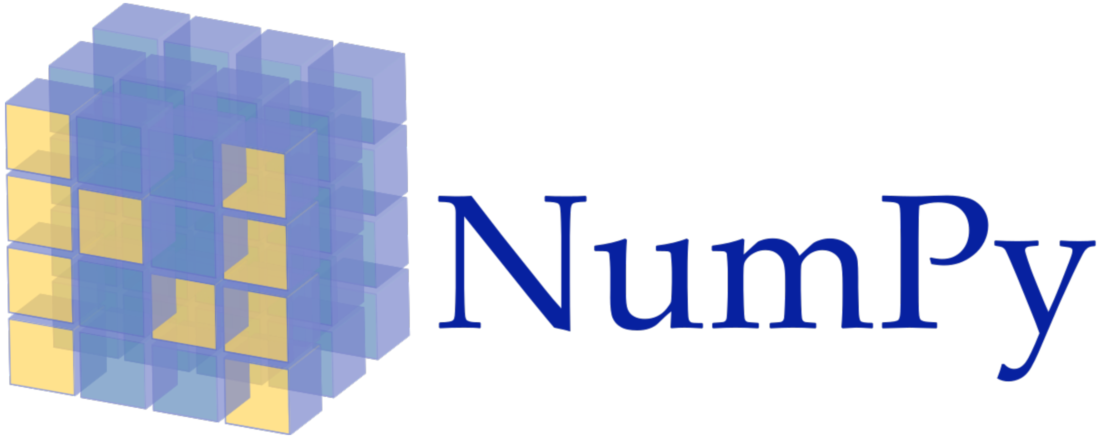
The Computer Language Benchmarks Game

9 / 63

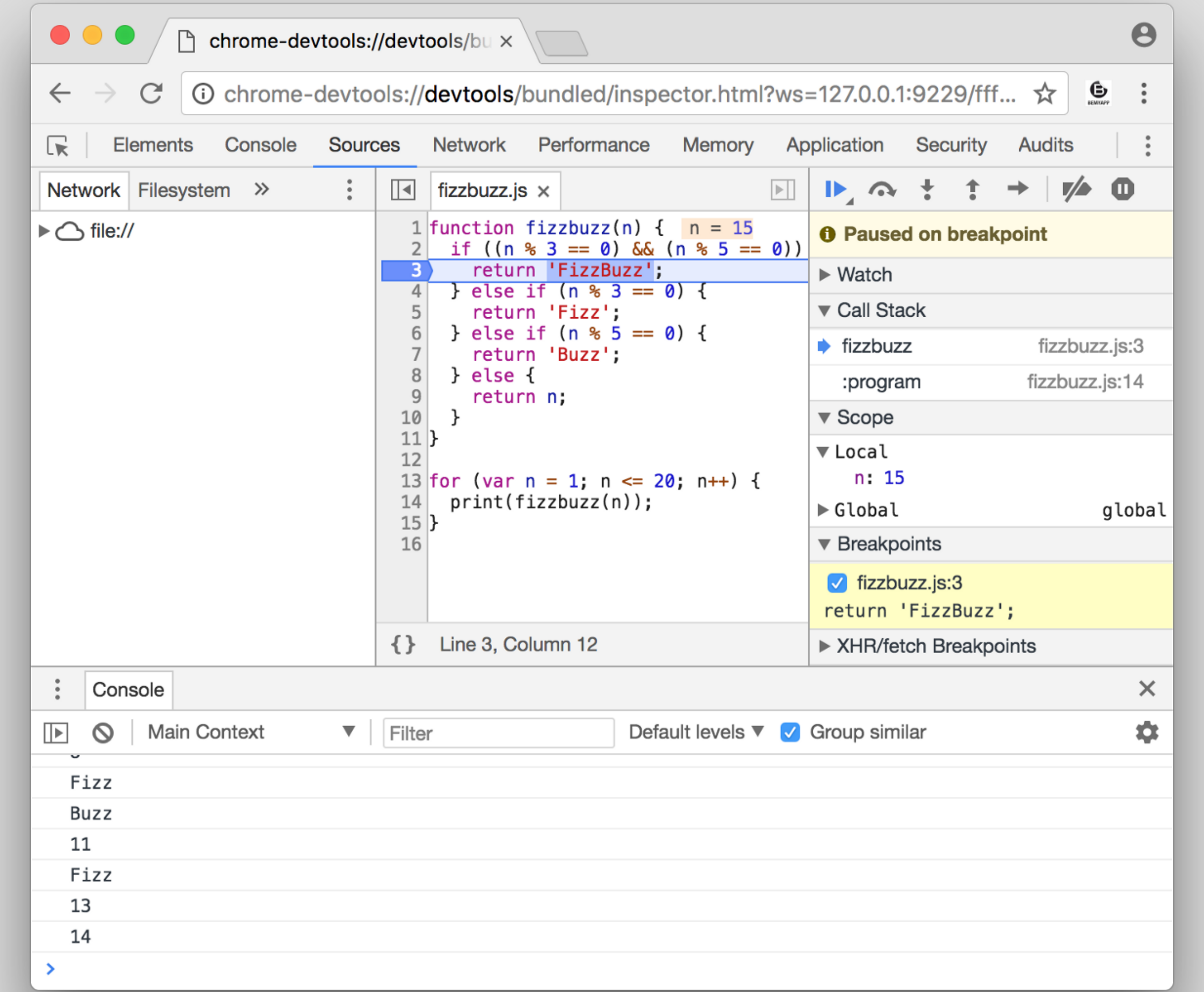
https://www.youtube.com/watch?v=mRKjWrNJ8DI



https://medium.com/graalvm/faster-r-with-fastr-4b8db0e0dceb


NumPy

SciPy



```
function fizzbuzz(n) {  n = 15
  if ((n % 3 == 0) && (n % 5 == 0)) {
    return 'FizzBuzz';
  } else if (n % 3 == 0) {
    return 'Fizz';
  } else if (n % 5 == 0) {
    return 'Buzz';
  } else {
    return n;
  }
}

for (var n = 1; n <= 20; n++) {
  print(fizzbuzz(n));
}
```

{ } Line 3, Column 12

Paused on breakpoint

▶ Watch

▼ Call Stack

fizzbuzz	fizzbuzz.js:3
:program	fizzbuzz.js:14

▼ Scope

```
▼ Local
    n: 15
▶ Global                   global
```

▼ Breakpoints

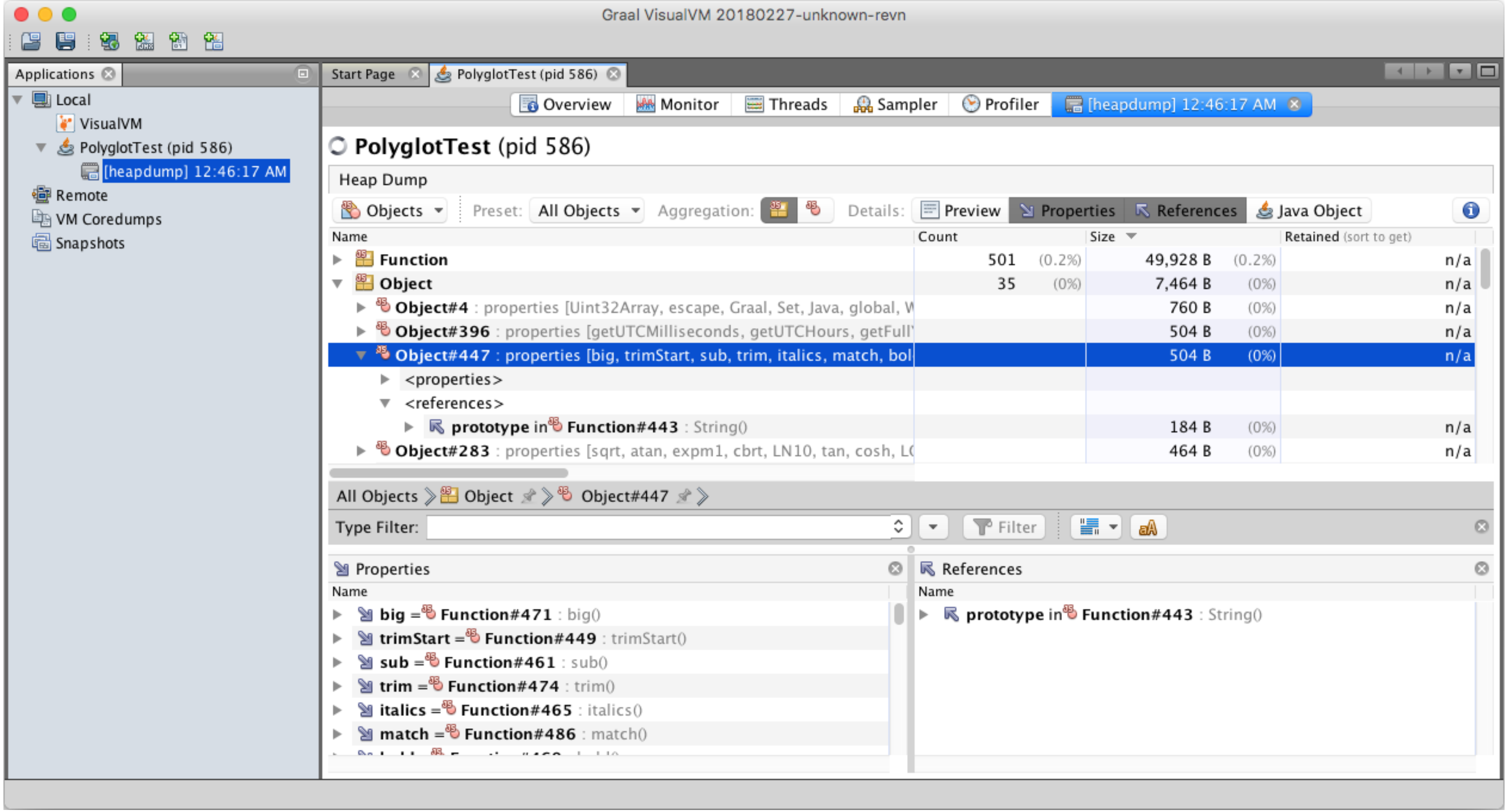
☑ fizzbuzz.js:3

```
return 'FizzBuzz';
```

▶ XHR/fetch Breakpoints

☑ Group similar

```
Fizz
Buzz
11
Fizz
13
14
>
```



Graal VisualVM 20180227-unknown-revn

Applications

- Local
  - VisualVM
  - PolyglotTest (pid 586)
    - [heapdump] 12:46:17 AM
- Remote
- VM Coredumps
- Snapshots

Start Page | PolyglotTest (pid 586)

Overview | Monitor | Threads | Sampler | Profiler | [heapdump] 12:46:17 AM

PolyglotTest (pid 586)

Heap Dump

Objects | Preset: All Objects | Aggregation: | Details: Preview | Properties | References | Java Object

Name	Count		Size		Retained (sort to get)
▶ Function	501	(0.2%)	49,928 B	(0.2%)	n/a
▼ Object	35	(0%)	7,464 B	(0%)	n/a
▶ Object#4 : properties [Uint32Array, escape, Graal, Set, Java, global, W			760 B	(0%)	n/a
▶ Object#396 : properties [getUTCMilliseconds, getUTCHours, getFull'			504 B	(0%)	n/a
▼ Object#447 : properties [big, trimStart, sub, trim, italics, match, bol			504 B	(0%)	n/a
▶ <properties>					
▼ <references>					
▶ prototype in Function#443 : String()			184 B	(0%)	n/a
▶ Object#283 : properties [sqrt, atan, expm1, cbrt, LN10, tan, cosh, LC			464 B	(0%)	n/a

All Objects > Object > Object#447 >

Type Filter: | Filter

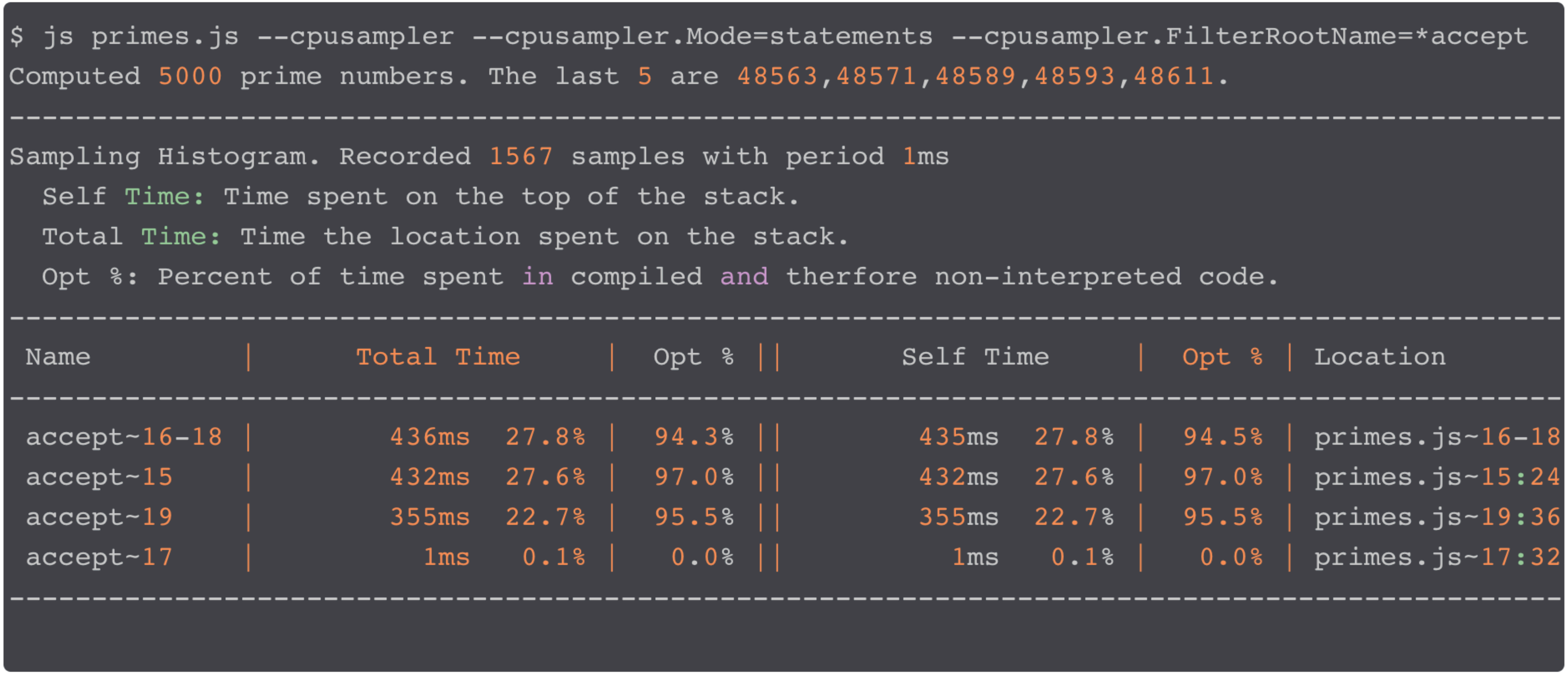
Properties

Name
▶ big = Function#471 : big()
▶ trimStart = Function#449 : trimStart()
▶ sub = Function#461 : sub()
▶ trim = Function#474 : trim()
▶ italics = Function#465 : italics()
▶ match = Function#486 : match()

References

Name
▶ prototype in Function#443 : String()

https://medium.com/graalvm/analyzing-the-heap-of-graalvm-polyglot-applications-b9963e68a6a

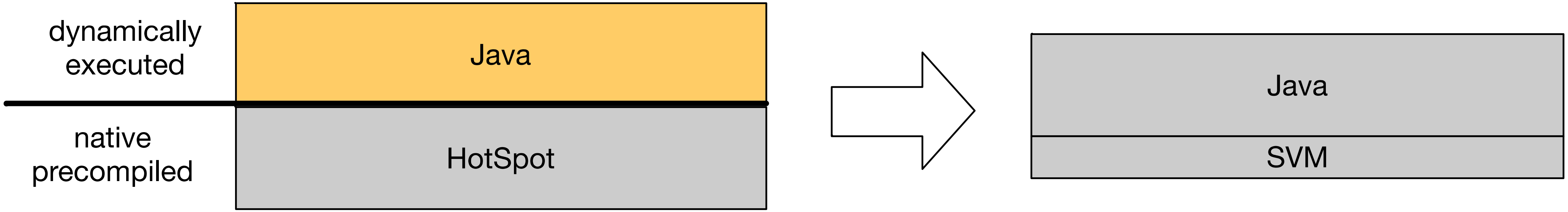


```
$ js primes.js --cpusampler --cpusampler.Mode=statements --cpusampler.FilterRootName=*accept
Computed 5000 prime numbers. The last 5 are 48563,48571,48589,48593,48611.
---------------------------------------------------------------------------------------------
Sampling Histogram. Recorded 1567 samples with period 1ms
  Self Time: Time spent on the top of the stack.
  Total Time: Time the location spent on the stack.
  Opt %: Percent of time spent in compiled and therfore non-interpreted code.
---------------------------------------------------------------------------------------------
 Name         |      Total Time     |  Opt % ||       Self Time     |  Opt % | Location
---------------------------------------------------------------------------------------------
 accept~16-18 |        436ms  27.8% |  94.3% ||        435ms  27.8% |  94.5% | primes.js~16-18
 accept~15    |        432ms  27.6% |  97.0% ||        432ms  27.6% |  97.0% | primes.js~15:24
 accept~19    |        355ms  22.7% |  95.5% ||        355ms  22.7% |  95.5% | primes.js~19:36
 accept~17    |          1ms   0.1% |   0.0% ||          1ms   0.1% |   0.0% | primes.js~17:32
---------------------------------------------------------------------------------------------
```

spring
by Pivotal™

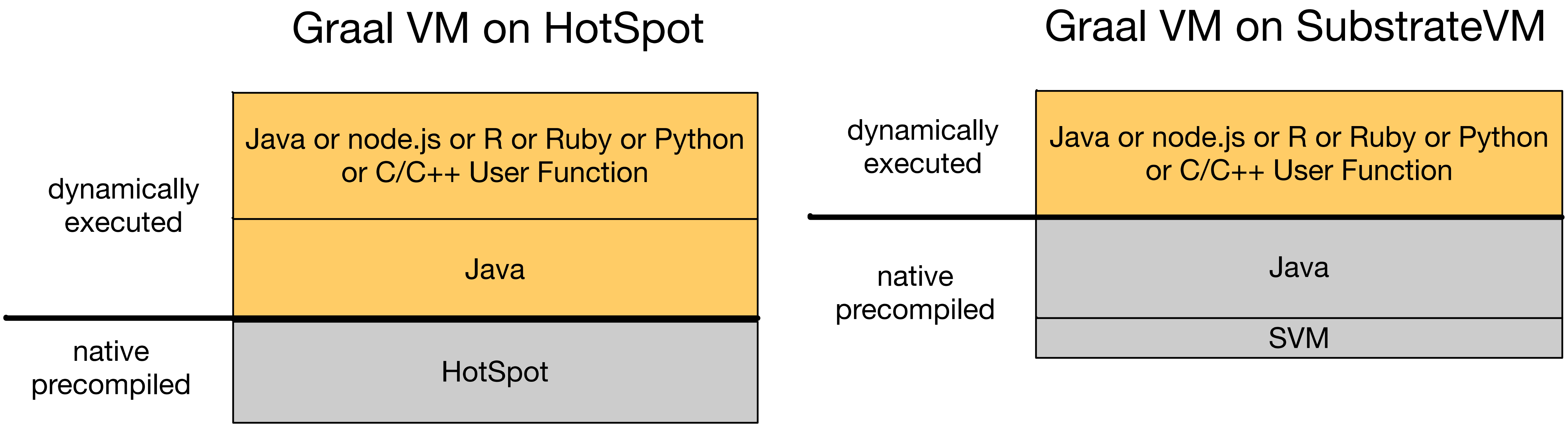
Native images

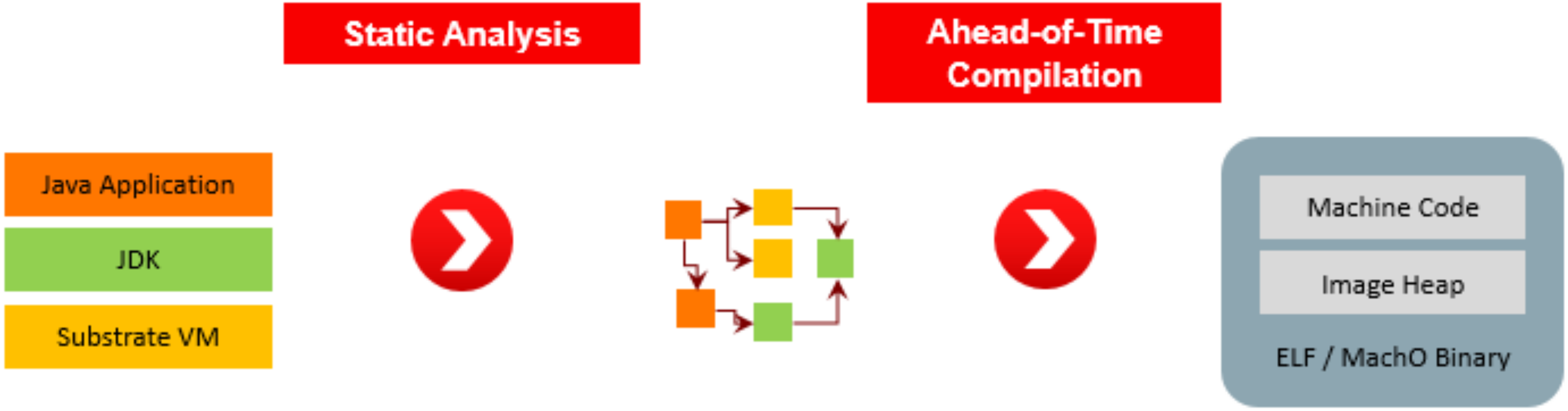
- Full AOT compilation to machine code
- Works with memory management
- Secure execution (e.g., bounds checks)
- **Embeddable with native applications**



Graal VM is a hybrid of static & dynamic runtimes

- Precompile core parts of application, but still allow extensibility!





Static Analysis

Ahead-of-Time Compilation

Java Application

JDK

Substrate VM

Machine Code

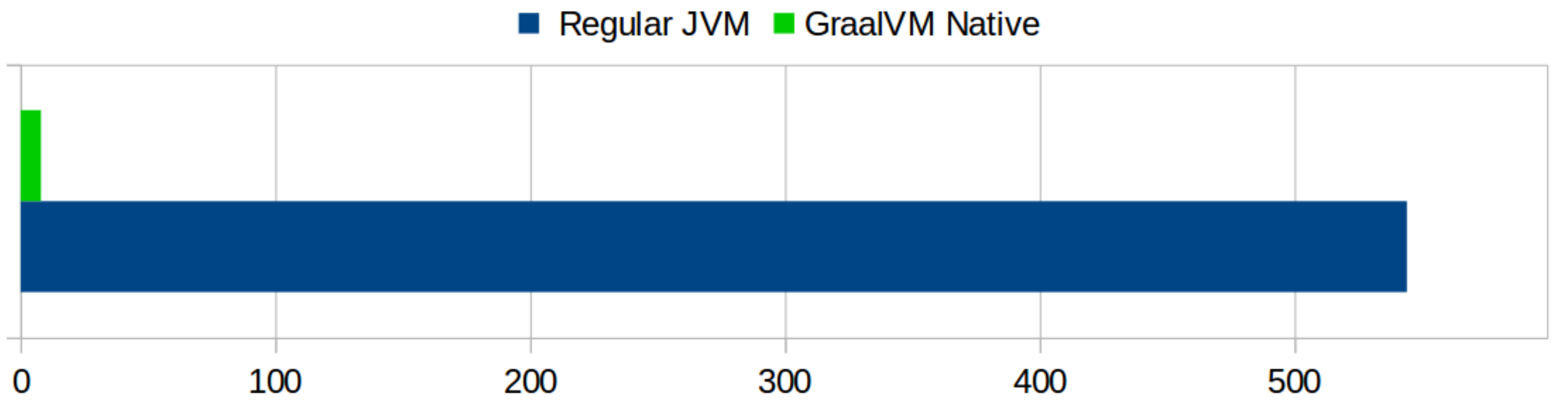
Image Heap

ELF / MachO Binary

All Java classes from Your application, JDK, and Substrate VM

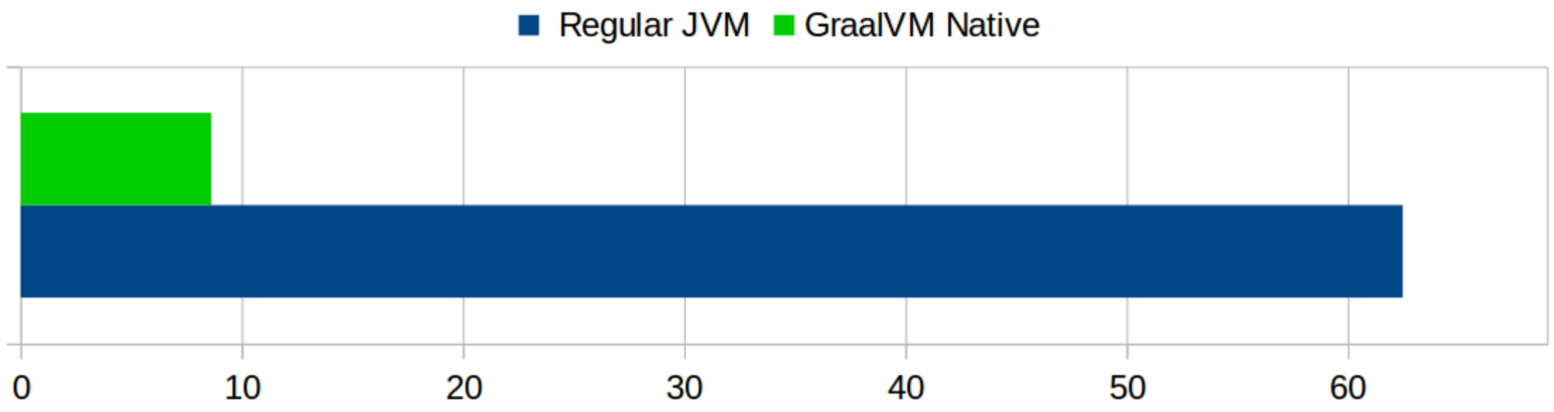
Reachable methods, fields, and classes

Application running without dependency on JDK and without Java class loading



Netty Startup Time

Real, wall clock time (milliseconds)

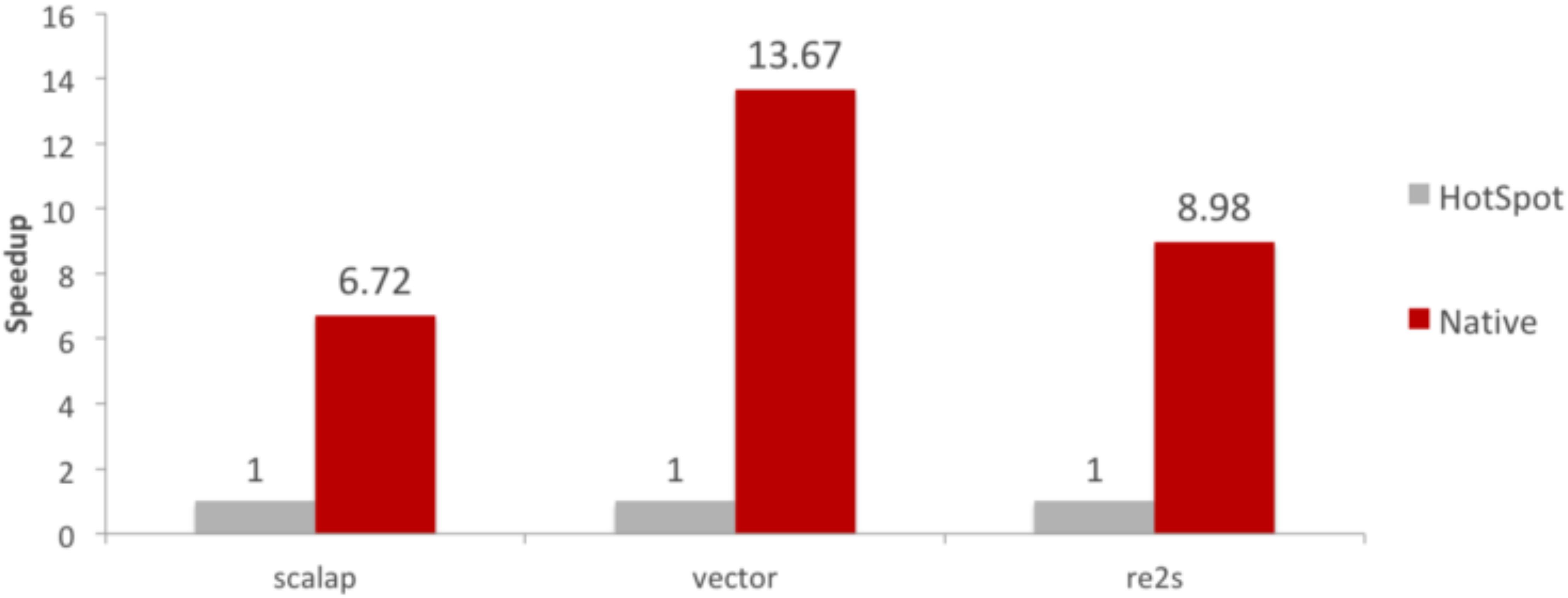


Netty Memory

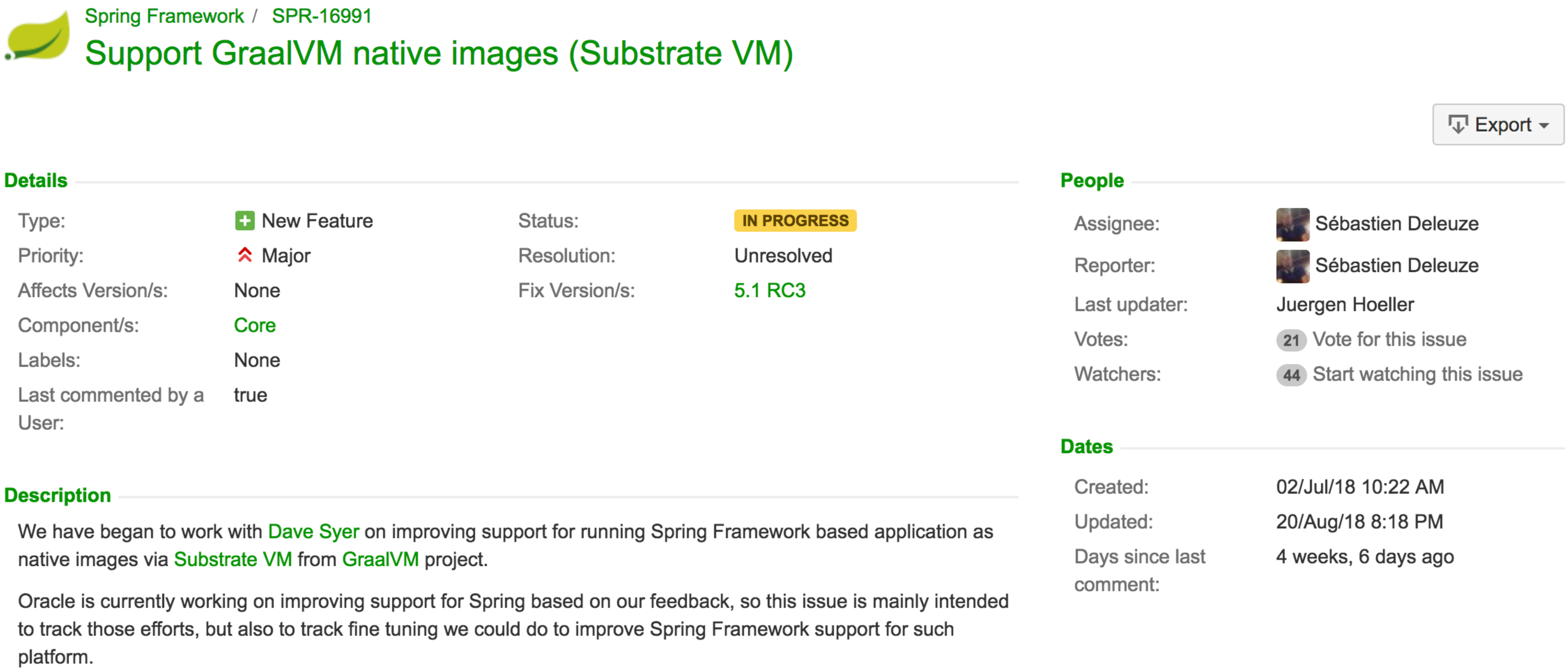
Maximum resident set size (MB)

Native scalac

```
git clone https://github.com/graalvm/graalvm-demos

cd graalvm-demos/scala-days-2018/scalac-native/scala-substitutions

sbt package

cd ../


$GRAALVM_HOME/bin/native-image -cp $SCALA_HOME/lib/scala-compiler.jar:
$SCALA_HOME/lib/scala-library.jar:$SCALA_HOME/lib/scala-reflect.jar:$PWD/
scalac-substitutions/target/scala-2.12/scalac-substitutions_2.12-0.1.0-
SNAPSHOT.jar \

  -H:SubstitutionResources=substitutions.json,substitutions-2.12.json \

  -H:ReflectionConfigurationFiles=scalac-substitutions/reflection-
config.json \

  -H:Class=scala.tools.nsc.Main \

  -H:Name=scalac
```



https://medium.com/graalvm/compiling-scala-faster-with-graalvm-86c5c0857fa3



Spring Framework / SPR-16991

Support GraalVM native images (Substrate VM)

Details

Type:	New Feature	Status:	IN PROGRESS
Priority:	Major	Resolution:	Unresolved
Affects Version/s:	None	Fix Version/s:	5.1 RC3
Component/s:	Core		
Labels:	None		
Last commented by a User:	true		

People

Assignee:	Sébastien Deleuze
Reporter:	Sébastien Deleuze
Last updater:	Juergen Hoeller
Votes:	21 Vote for this issue
Watchers:	44 Start watching this issue

Dates

Created:	02/Jul/18 10:22 AM
Updated:	20/Aug/18 8:18 PM
Days since last comment:	4 weeks, 6 days ago

Description

We have began to work with Dave Syer on improving support for running Spring Framework based application as native images via Substrate VM from GraalVM project.

Oracle is currently working on improving support for Spring based on our feedback, so this issue is mainly intended to track those efforts, but also to track fine tuning we could do to improve Spring Framework support for such platform.

https://jira.spring.io/browse/SPR-16991

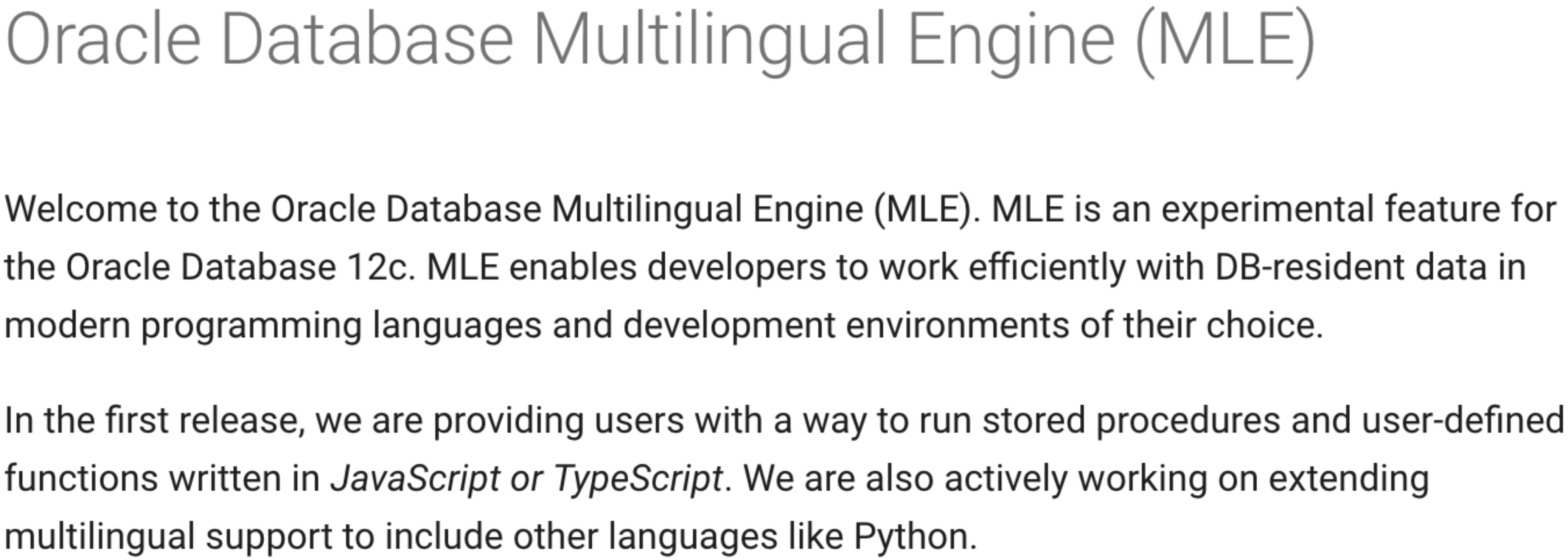
Java™
11

GraalVM™

High performance, polyglot, language-level virtualization layer…

embeddable across the stack

in native and JVM-based applications.

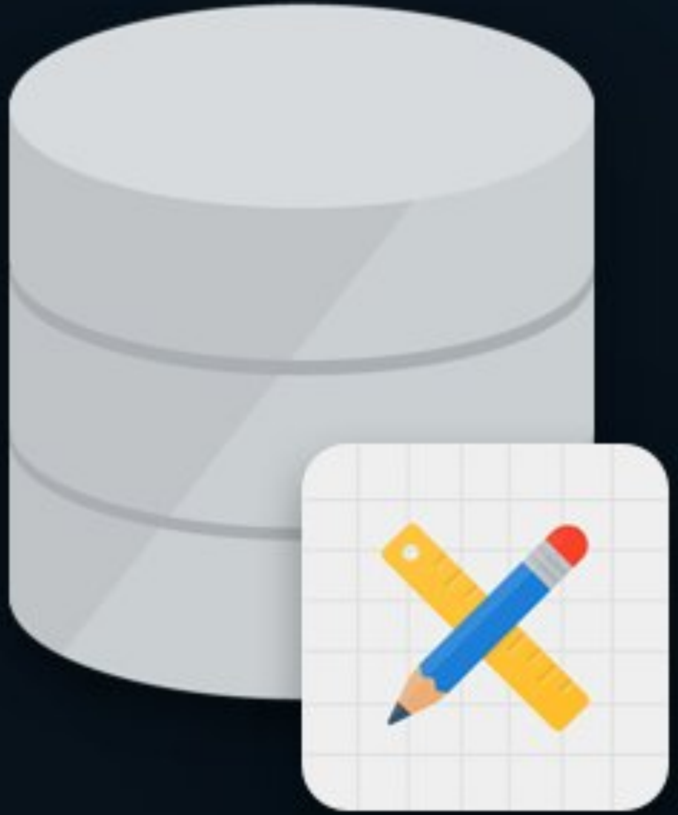


Oracle Database Multilingual Engine (MLE)

Welcome to the Oracle Database Multilingual Engine (MLE). MLE is an experimental feature for the Oracle Database 12c. MLE enables developers to work efficiently with DB-resident data in modern programming languages and development environments of their choice.

In the first release, we are providing users with a way to run stored procedures and user-defined functions written in *JavaScript or TypeScript*. We are also actively working on extending multilingual support to include other languages like Python.

https://oracle.github.io/oracle-db-mle/releases/0.2.7/



Oracle Database +
APEX +
JavaScript/Python =

Awesome!

https://apexea.oracle.com

https://blogs.oracle.com/apex/oracle-database-%2b-apex-%2b-javascriptpython-%3d-awesome

Team

Oracle
Florian Angerer
Danilo Ansaloni
Stefan Anzinger
Martin Balin
Cosmin Basca
Daniele Bonetta
Dušan Bálek
Matthias Brantner
Lucas Braun
Petr Chalupa
Jürgen Christ
Laurent Daynès
Gilles Duboscq
Svatopluk Dědic
Martin Entlicher
Pit Fender
Francois Farquet
Brandon Fish
Matthias Grimmer
Christian Häubl
Peter Hofer
Bastian Hossbach
Christian Humer
Tomáš Hůrka
Mick Jordan

Oracle (continued)
Vojin Jovanovic
Anantha Kandukuri
Harshad Kasture
Cansu Kaynak
Peter Kessler
Duncan MacGregor
Jiří Maršík
Kevin Menard
Miloslav Metelka
Tomáš Myšík
Petr Pišl
Oleg Pliss
Jakub Podlešák
Aleksandar Prokopec
Tom Rodriguez
Roland Schatz
Benjamin Schlegel
Chris Seaton
Jiří Sedláček
Doug Simon
Štěpán Šindelář
Zbyněk Šlajchrt
Boris Spasojevic
Lukas Stadler
Codrut Stancu

Oracle (continued)
Jan Štola
Tomáš Stupka
Farhan Tauheed
Jaroslav Tulach
Alexander Ulrich
Michael Van De Vanter
Aleksandar Vitorovic
Christian Wimmer
Christian Wirth
Paul Wögerer
Mario Wolczko
Andreas Wöß
Thomas Würthinger
Tomáš Zezula
Yudi Zheng

Red Hat
Andrew Dinn
Andrew Haley

Intel
Michael Berg

Twitter
Chris Thalinger

Oracle Interns
Brian Belleville
Ondrej Douda
Juan Fumero
Miguel Garcia
Hugo Guiroux
Shams Imam
Berkin Ilbeyi
Hugo Kapp
Alexey Karyakin
Stephen Kell
Andreas Kunft
Volker Lanting
Gero Leinemann
Julian Lettner
Joe Nash
Tristan Overney
Aleksandar Pejovic
David Piorkowski
Philipp Riedmann
Gregor Richards
Robert Seilbeck
Rifat Shariyar

Oracle Alumni
Erik Eckstein
Michael Haupt
Christos Kotselidis
David Leibs
Adam Welc
Till Westmann

JKU Linz
Hanspeter Mössenböck
Benoit Daloze
Josef Eisl
Thomas Feichtinger
Josef Haider
Christian Huber
David Leopoldseder
Stefan Marr
Manuel Rigger
Stefan Rumzucker
Bernhard Urban

TU Berlin:
Volker Markl
Andreas Kunft
Jens Meiners
Tilmann Rabl

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

LaBRI
Floréal Morandat

University of California, Irvine
Michael Franz
Yeoul Na
Mohaned Qunaibit
Gulfem Savrun Yeniceri
Wei Zhang

Purdue University
Jan Vitek
Tomas Kalibera
Petr Maj
Lei Zhao

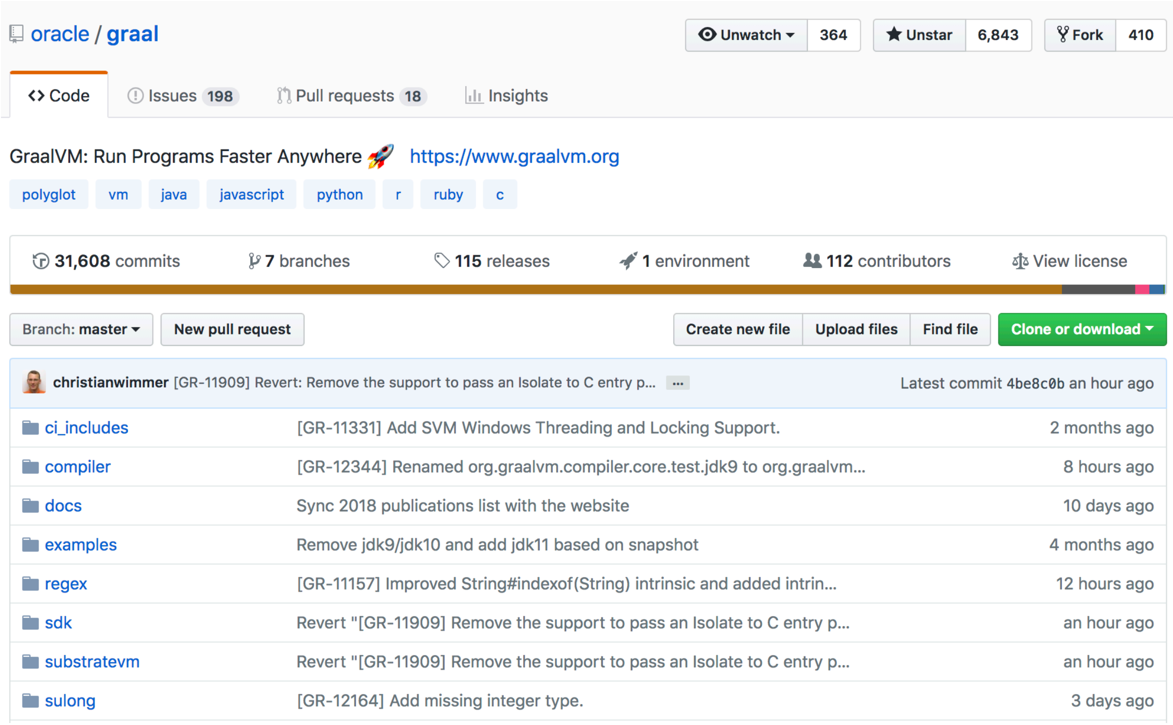
T. U. Dortmund
Peter Marwedel
Helena Kotthaus
Ingo Korb

University of California, Davis
Duncan Temple Lang
Nicholas Ulle

University of Lugano, Switzerland
Walter Binder
Sun Haiyang

Building a Universal VM is a Community Effort

- Test your applications with GraalVM
  - Documentation and downloads at http://www.graalvm.org
- Connect your technology with GraalVM
  - Integrate GraalVM into your application
  - Run your own programming language or DSL
  - Build language-agnostic tools
- Join the conversation
  - Report issues or pull requests on GitHub
  - graalvm-users@oss.oracle.com
  - Follow @graalvm



oracle / graal

GraalVM: Run Programs Faster Anywhere 🚀 https://www.graalvm.org

31,608 commits · 7 branches · 115 releases · 1 environment · 112 contributors · View license

christianwimmer [GR-11909] Revert: Remove the support to pass an Isolate to C entry p... … Latest commit 4be8c0b an hour ago

ci_includes	[GR-11331] Add SVM Windows Threading and Locking Support.	2 months ago
compiler	[GR-12344] Renamed org.graalvm.compiler.core.test.jdk9 to org.graalvm...	8 hours ago
docs	Sync 2018 publications list with the website	10 days ago
examples	Remove jdk9/jdk10 and add jdk11 based on snapshot	4 months ago
regex	[GR-11157] Improved String#indexof(String) intrinsic and added intrin...	12 hours ago
sdk	Revert "[GR-11909] Remove the support to pass an Isolate to C entry p...	an hour ago
substratevm	Revert "[GR-11909] Remove the support to pass an Isolate to C entry p...	an hour ago
sulong	[GR-12164] Add missing integer type.	3 days ago

Please

Remember to rate this session

Thank you!

@GOTOber

gotober.com