

ORACLE

# GraalVM History, Status, Vision

Thomas Wuerthinger  
GraalVM Founder & Project Lead  
@thomaswue

25-November-2019

© 2019 Oracle

# Origins of GraalVM

---

- HotSpot C1 (C++) transformed into C1X in Maxine Research VM (Java)
- C1X made runtime independent and plugged into HotSpot (C1X4HotSpot)
- High-level IR redesign together with JKU Linz PhD students Lukas Stadler and Gilles Duboscq
- Graal project in OpenJDK including Doug Simon and Christian Wimmer
- Truffle to also execute JavaScript; JKU Linz students Andreas Woess and Christian Humer
- Adding FastR project (data base integration) and TruffleRuby (Chris Seaton)
- SubstrateVM (later "native image") for efficient database integration
- GraalVM as umbrella project for Graal-based technologies

# One VM to Rule Them All

Thomas Würthinger\* Christian Wimmer\* Andreas Wöß† Lukas Stadler†  
Gilles Duboscq† Christian Humer† Gregor Richards§ Doug Simon\* Mario Wolczko\*

\*Oracle Labs †Institute for System Software, Johannes Kepler University Linz, Austria §S<sup>3</sup> Lab, Purdue University  
{thomas.wuerthinger, christian.wimmer, doug.simon, mario.wolczko}@oracle.com  
{woess, stadler, duboscq, christian.humer}@ssw.jku.at gr@purdue.edu

# GraalVM will be the Universal VM

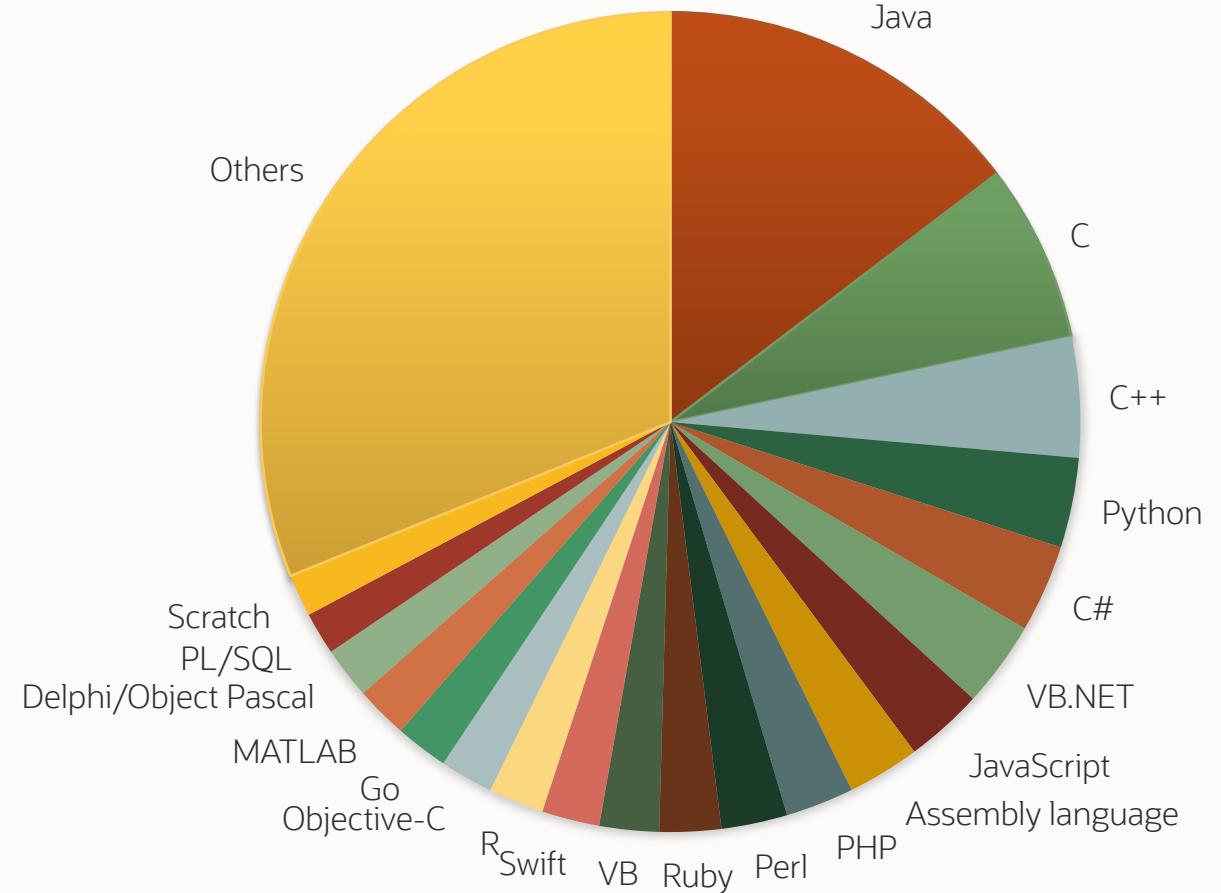
---

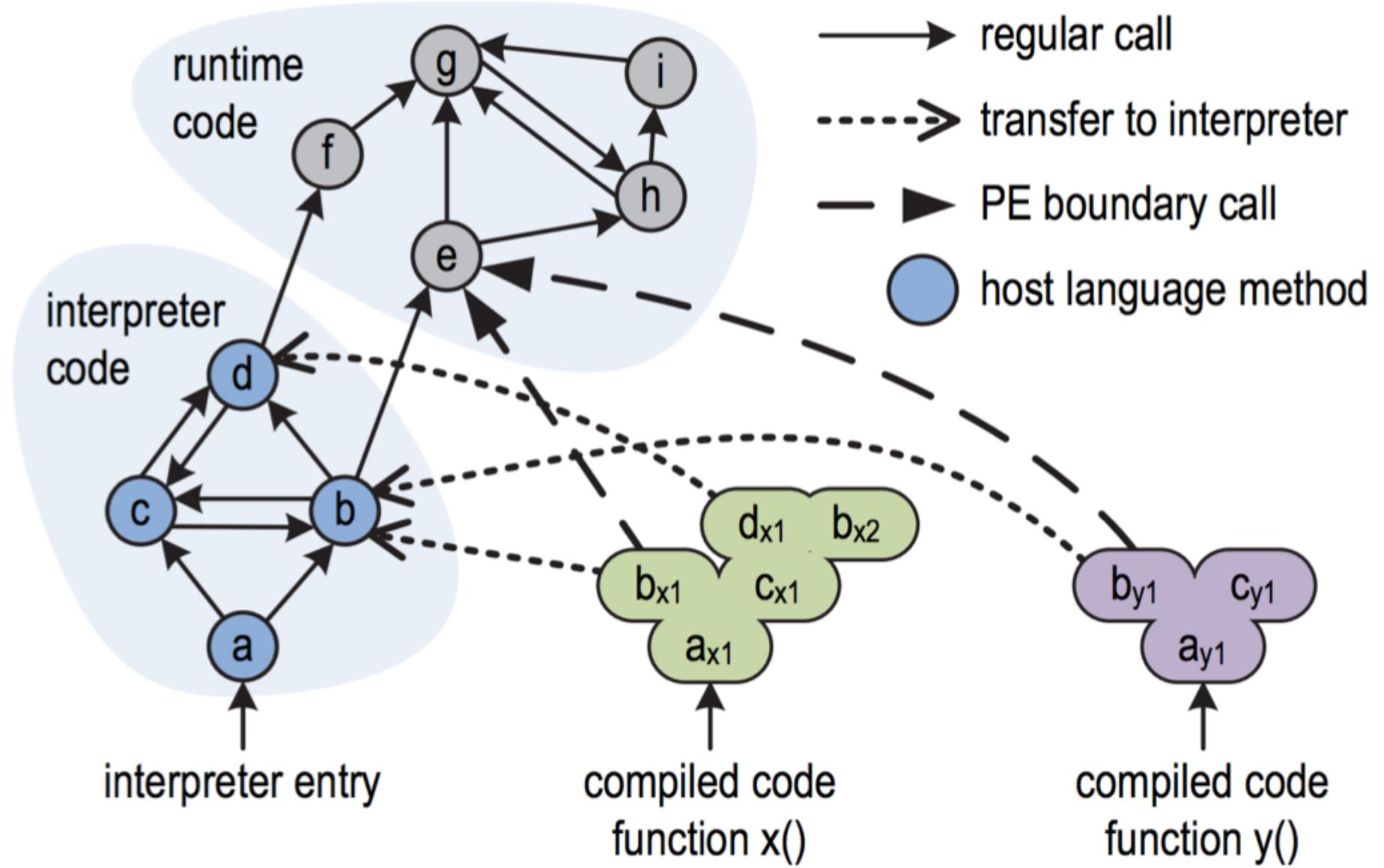
ubiquitous across the cloud stack

1. run programs more efficient
2. make developers more productive

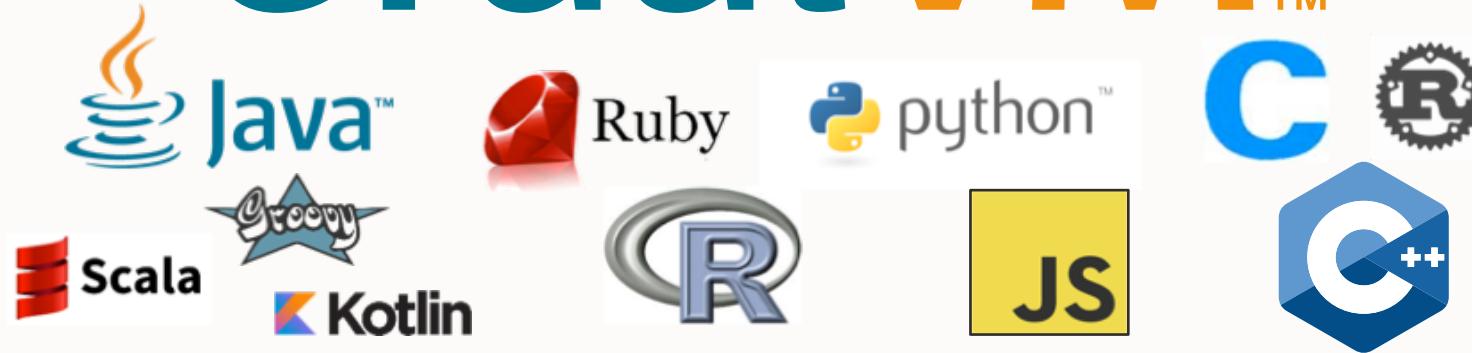


Programming Language Popularity  
(TOP 20 Languages From May 2018 Tiobe INDEX)





# GraalVM™



OpenJDK™



node  
js®



ORACLE®  
Database



Native Image



# What is GraalVM?

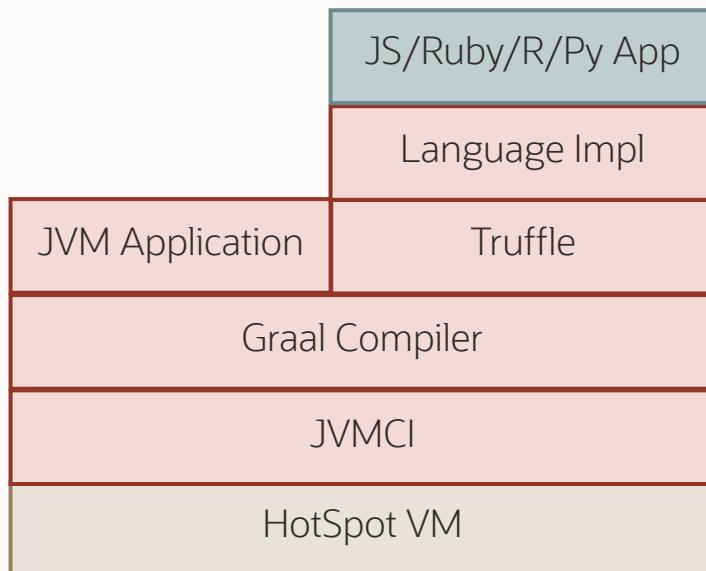
---

- Drop-in replacement for Java 8 and Java 11
  - Run your Java application faster
- Ahead-of-time compilation for Java
  - Create standalone binaries with low footprint
- High-performance JavaScript, Python, Ruby, R, ...
  - The first VM for true polyglot programming
  - Implement your own language or DSL

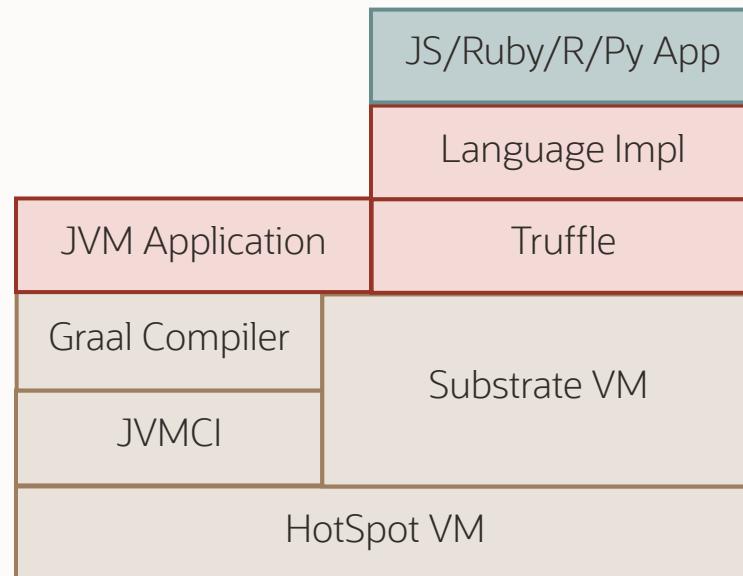
# Technical Architecture

---

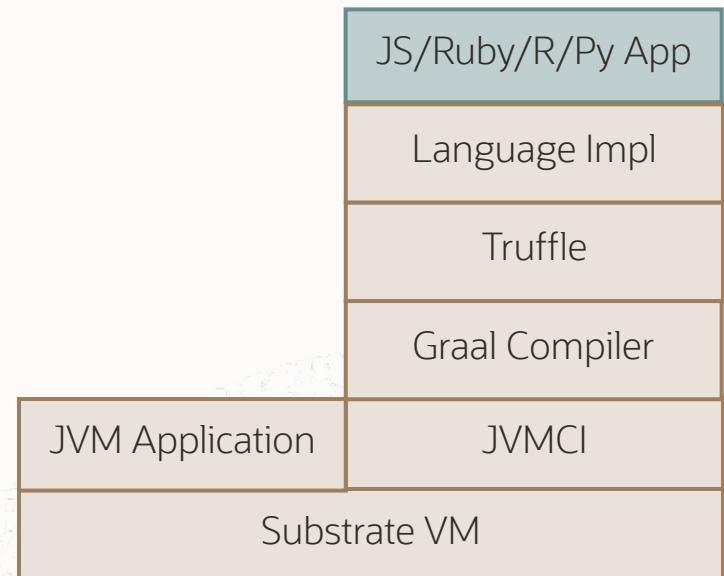
GraalVM JIT „jargraal“



GraalVM JIT „libgraal“

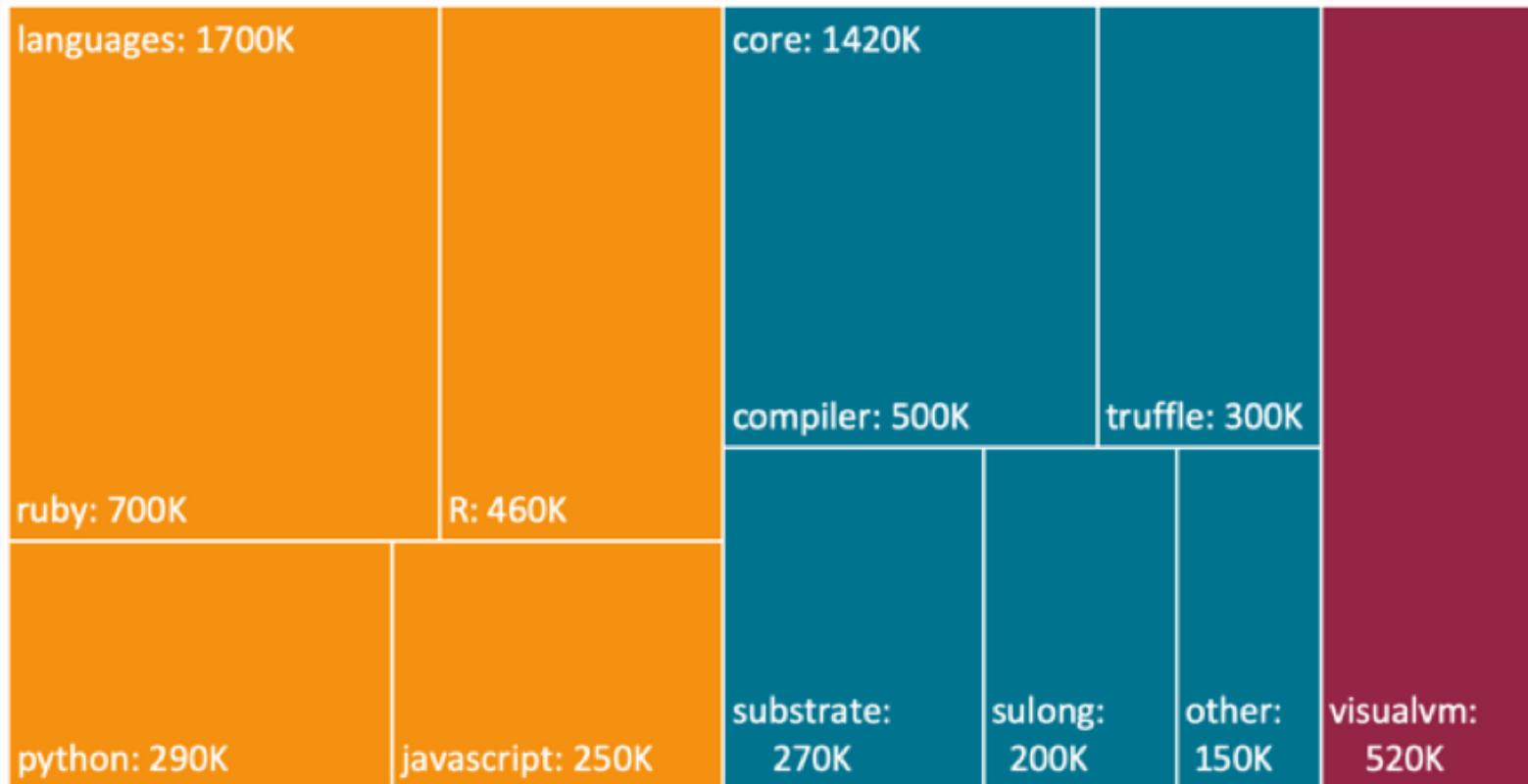


GraalVM AOT (native image)



# GraalVM Open Source

Open Source LOC actively maintained by GraalVM team



Total: 3,640,000 lines of code



**Chris Newland**  
@chriswhocodes

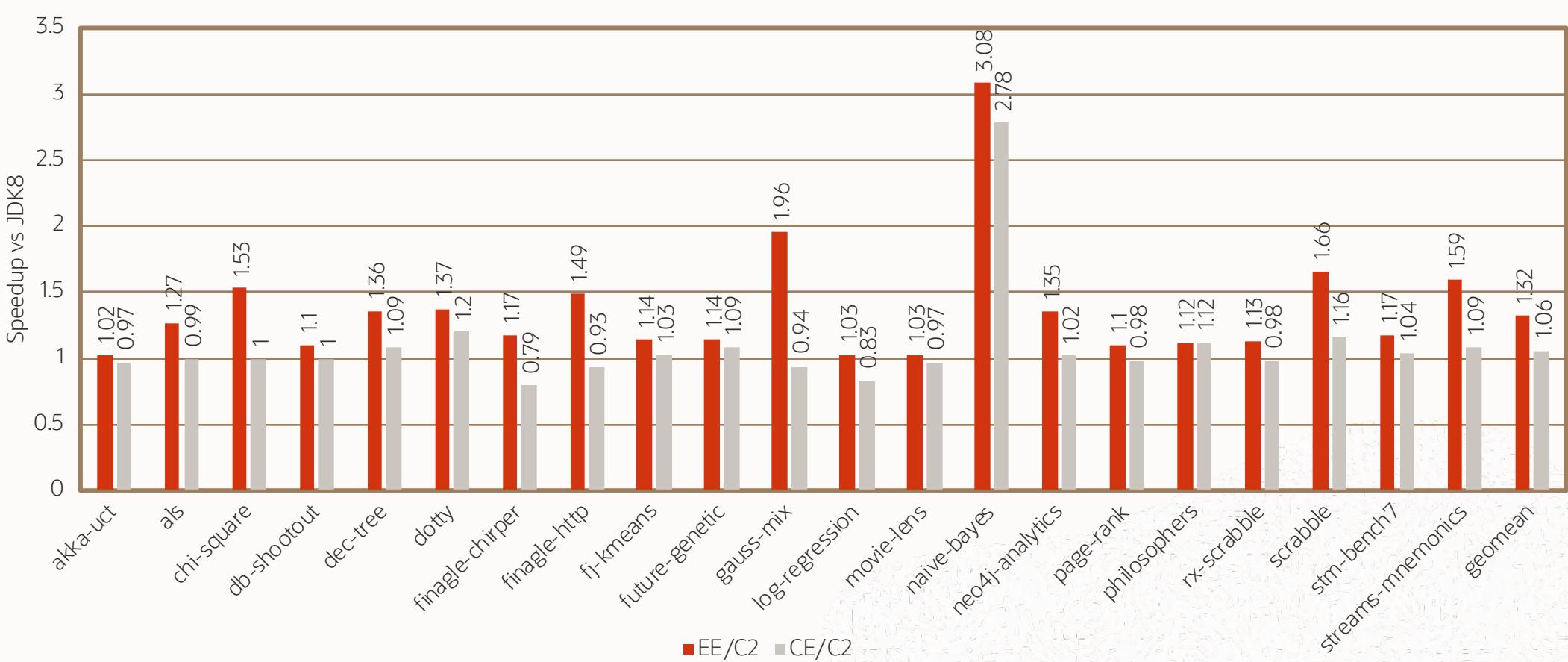
Replies to @mjpt777 @nipafx and 2 others

I tell my devs that hot code should look like it was written  
by my 9-year old ;) No functional or advanced OO!

12:00 PM · Aug 27, 2019 · Twitter for iPhone

GraalVM Vision:  
Abstractions should be without performance regret!

# GraalVM JIT Performance: Renaissance.dev



## More Benchmarks...

---

Optimizing for too few benchmarks is like overfitting a machine learning algorithm.

Therefore we started together with academic collaborators

<https://renaissance.dev>

We are looking for the community to contribute benchmarks for the frameworks they care about most.

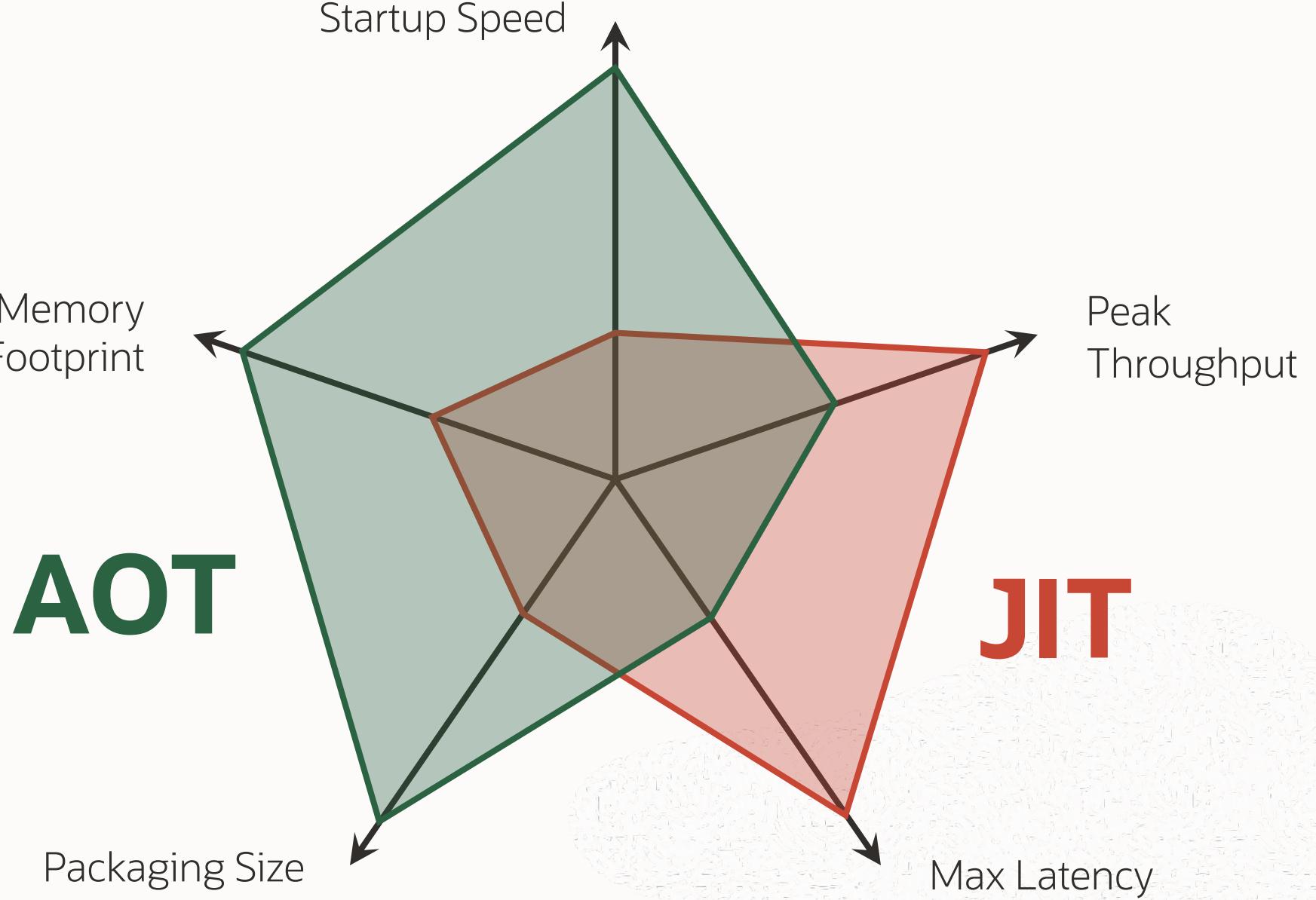


java MyMainClass  
**OpenJDK™**

native-image MyMainClass  
./mymainclass

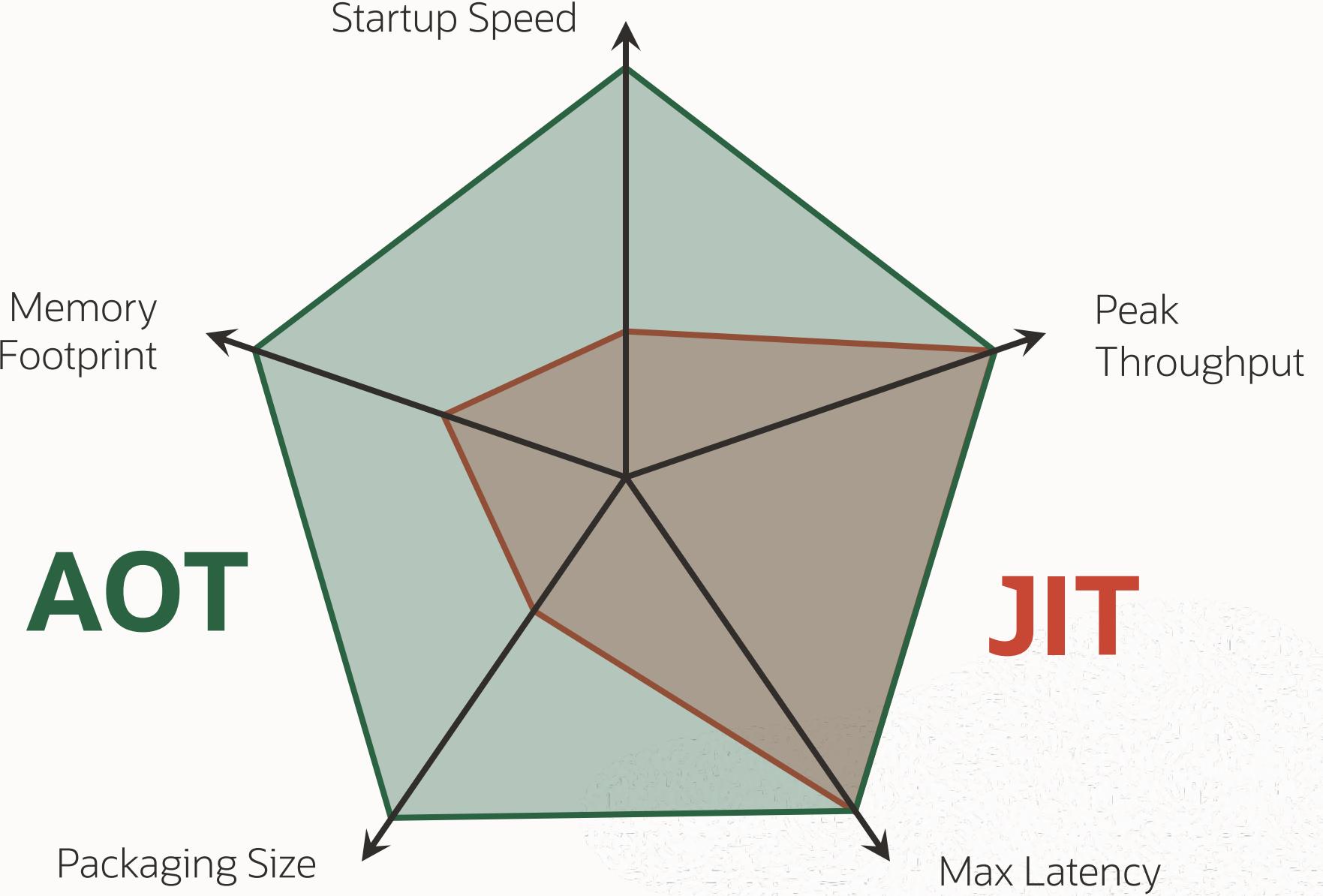


**Currently**

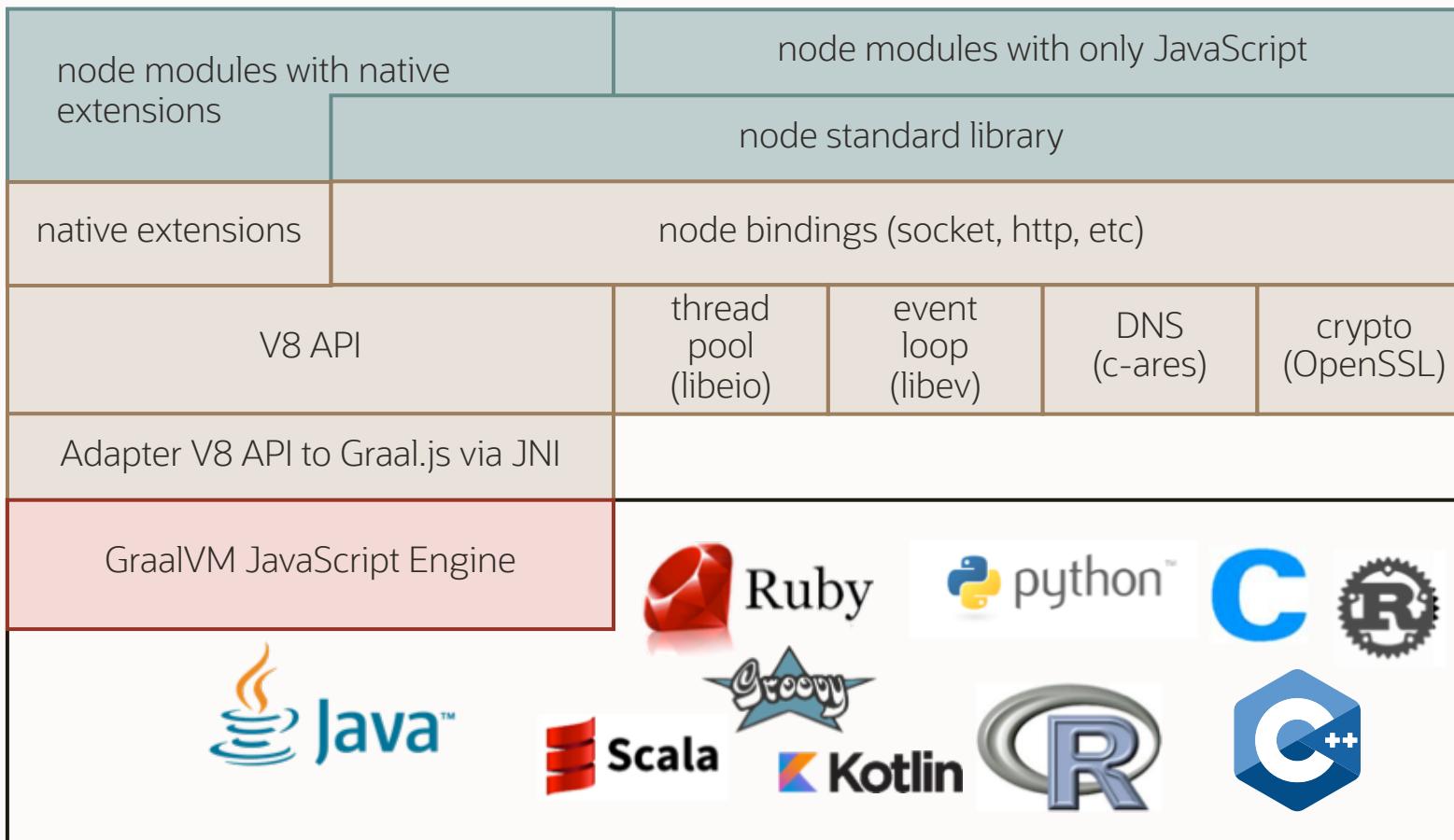


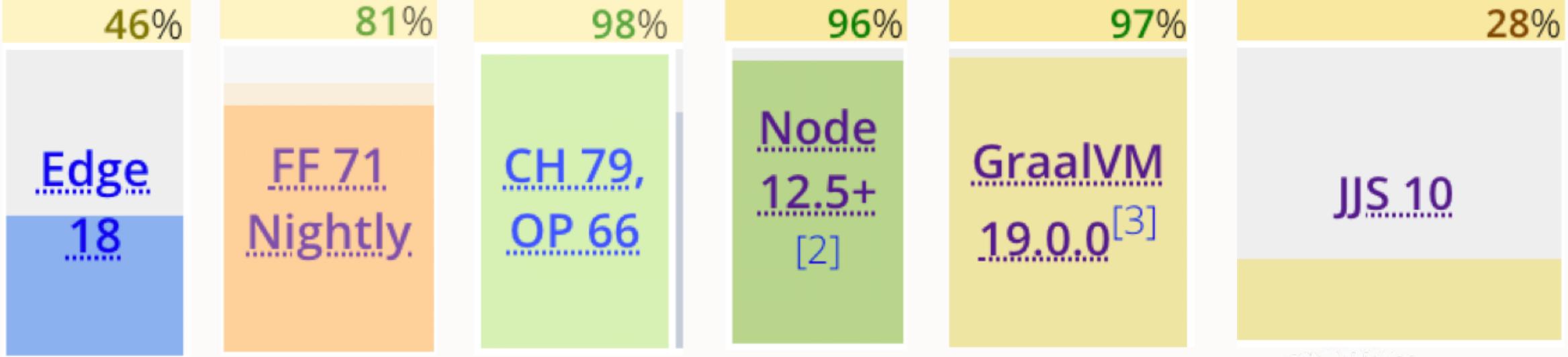
## Goal

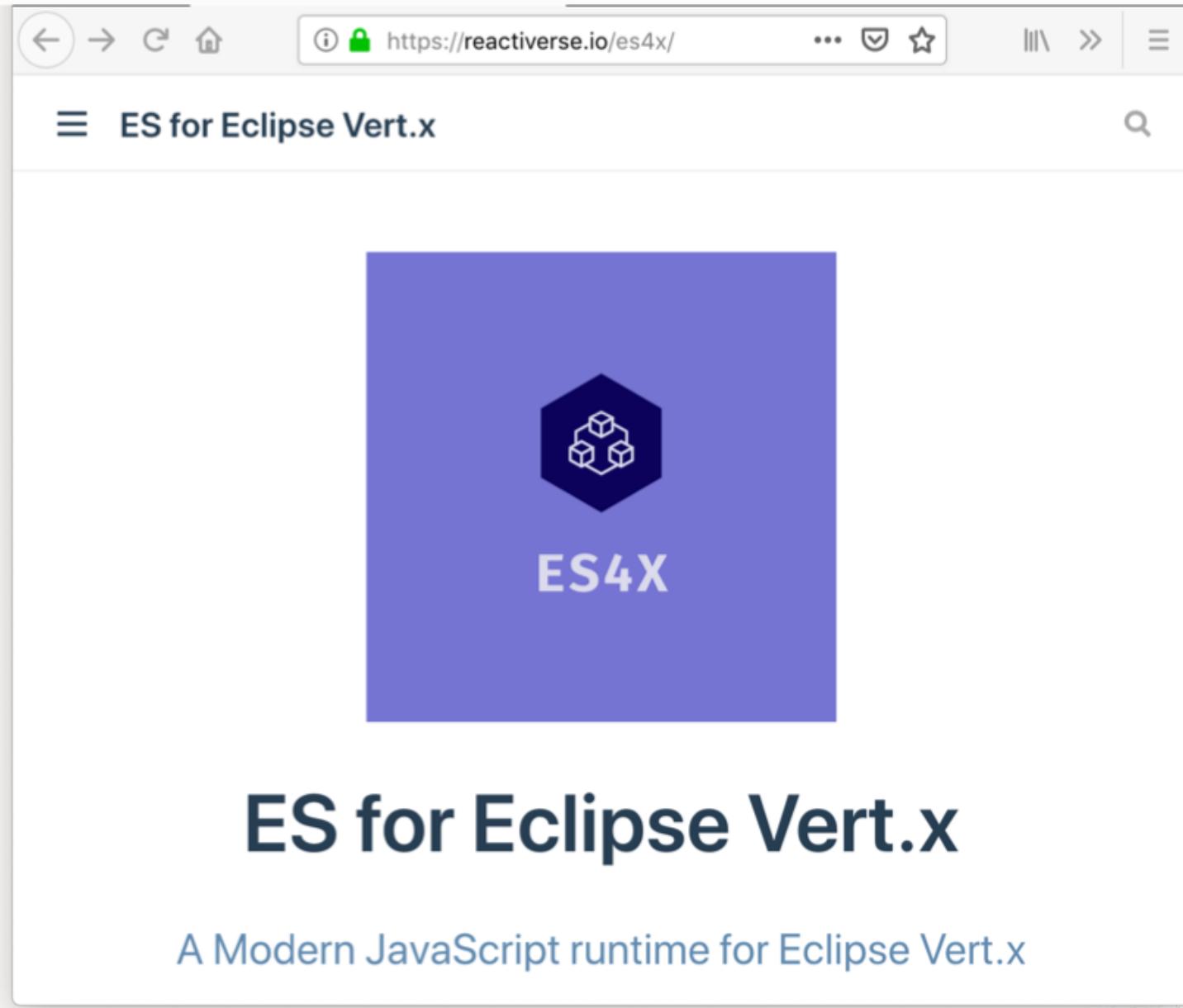
—



# Architecture of Node.js running via GraalVM







The screenshot shows a web browser window displaying the official website for ES for Eclipse Vert.x. The URL in the address bar is <https://reactiverse.io/es4x/>. The page has a light gray header with a dark blue navigation bar containing icons for back, forward, search, and other browser functions. The main content area features a large purple hexagonal logo with a white icon of three cubes and the text "ES4X" below it. To the right, there's a section titled "Performant" with the subtext "ES4X runs on top of GraalVM offering a great performance for JavaScript". The background of the page is a subtle, abstract pattern of interconnected nodes.

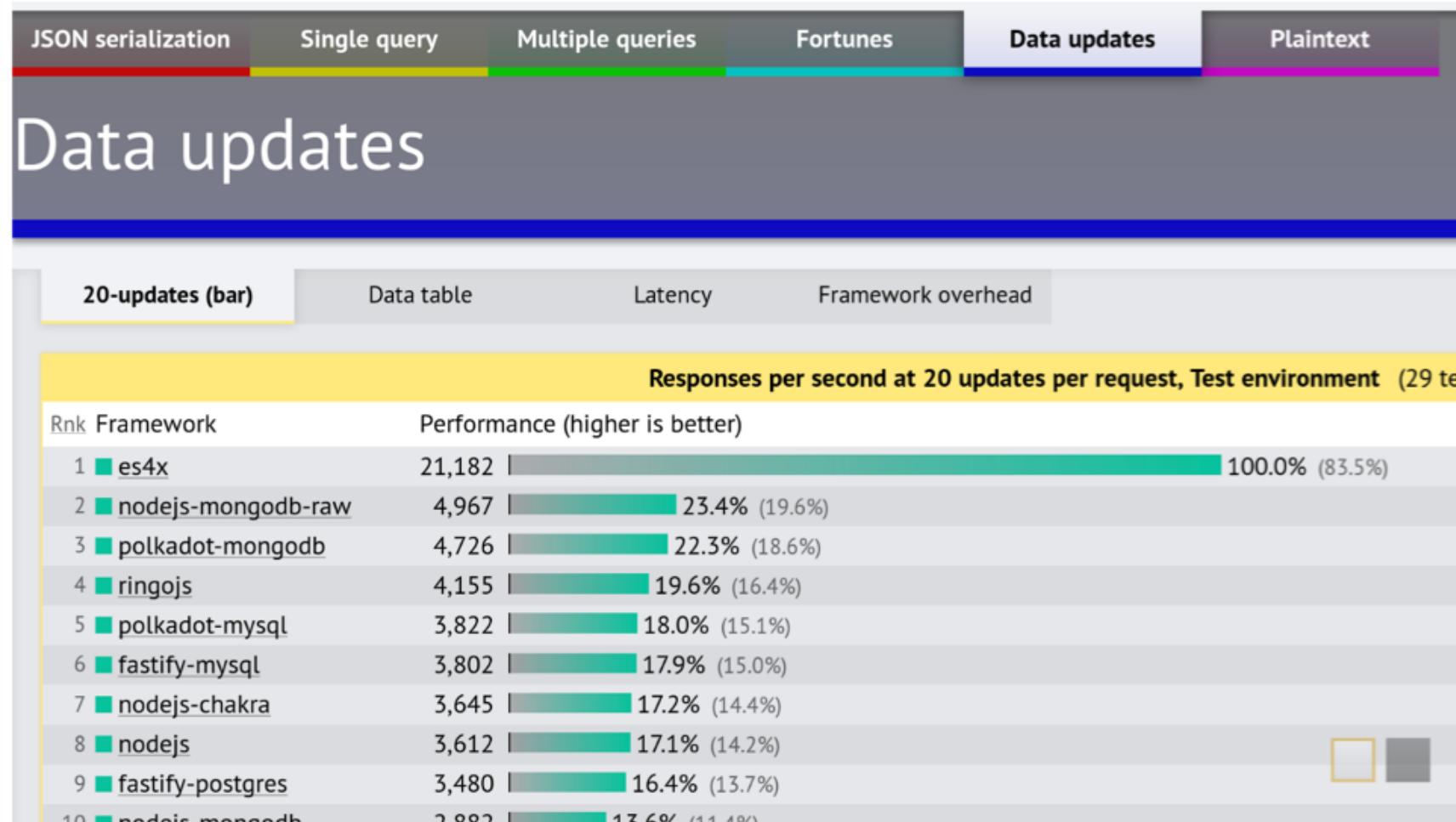
# ES for Eclipse Vert.x

A Modern JavaScript runtime for Eclipse Vert.x

# Performance

ES4X is the fastest [JavaScript](#) according to TechEmpower Frameworks Benchmark Round #18.

ES4X is the fastest on all tests when compared to [JavaScript](#) frameworks:



# FastR

---

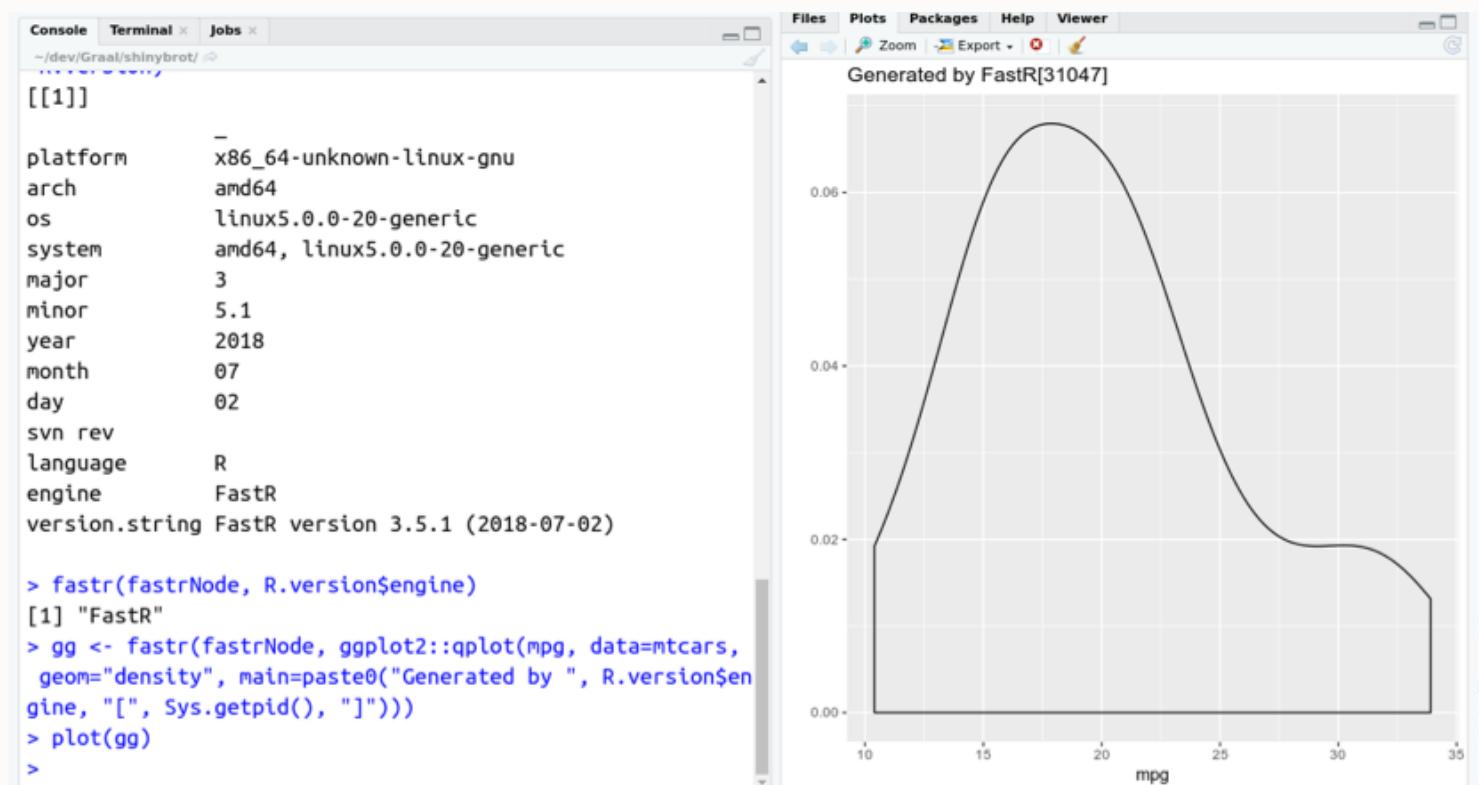
- GNU-R compatible R implementation
  - Including the C/Fortran interface
- Built on top of the GraalVM platform
  - Leverages GraalVM optimizing compiler
  - Integration with GraalVM dev tools
  - Zero overhead interop with other GraalVM languages

GraalVM™

# FastR Cluster Package

fastRCluster package to use FastR as a “cluster” from GNU-R

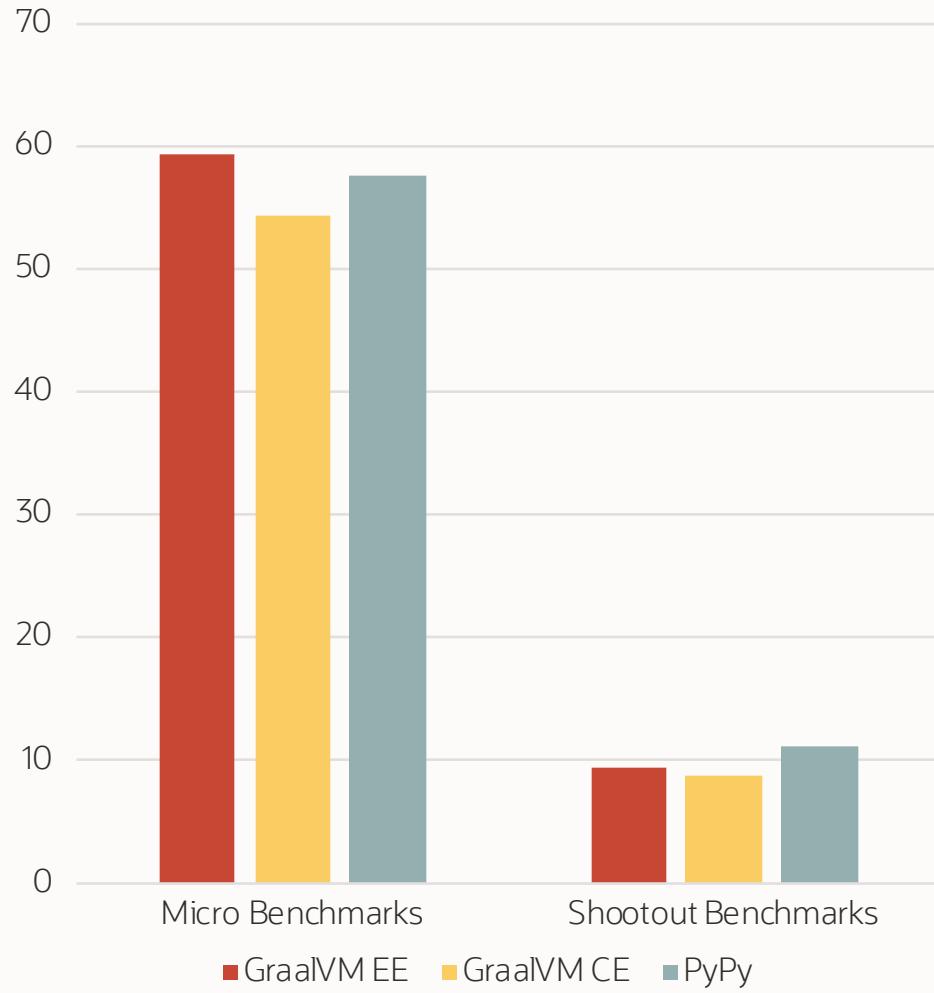
- Works like other “cluster” providers packages
- Offload to efficient R runtime similar to offloading to cluster



# Python Performance

Comparable to PyPy, the fastest alternative

Geomean Speedup over CPython  
(more is better)



# Using Numpy from Java – Calling into Numpy

```
try (Context context = Context.newBuilder()
    .option("python.PythonPath", "/path/to/numpy-1.16.4-py3.7-macosx-10.14-x86_64.egg")
    .allowAllAccess(true).build()) {
    Value geomean = context.eval("python", "import numpy\n" + "import math\n" +
        "lambda x: math.pow(numpy.array(x).prod(), 1/len(x))");
}

anonymous function that calculates the geometric
mean using numpy and the math module

double[] values = new double[] { 1, 5, 8, 3, 5, 8, 8, 7, 5, 6 };
double mean = geomean.execute(values).asDouble();
System.out.println(mean);
// 4.905181164183902

}
```

## GraalVM (pid 85381)

Graal Sampler

Sample:  CPU  Memory  Stop

Status: sampling inactive

CPU samples

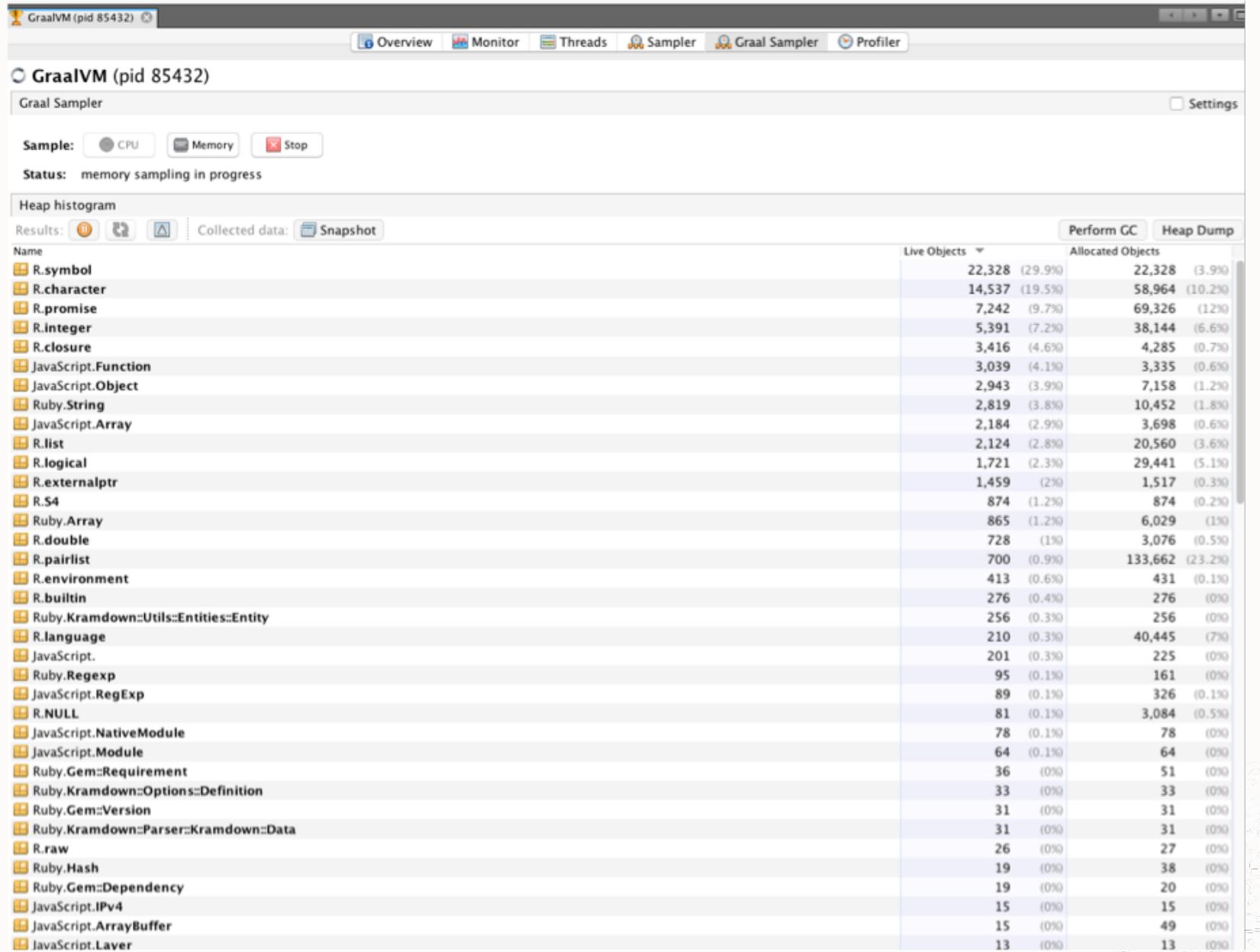
Results: View: Collected data: Snapshot

Name

Name	Total Time	Total Time (CPU)
main	70,314 ms (100%)	2,450 ms:
JavaScript.anonymous ()	70,314 ms (100%)	2,450 ms:
JavaScript.parserOnHeadersComplete ()	3,085 ms (4.4%)	2,235 ms:
JavaScript.parserOnIncoming ()	3,085 ms (4.4%)	2,235 ms:
JavaScript.emit ()	3,085 ms (4.4%)	2,235 ms:
JavaScript.emitTwo ()	3,085 ms (4.4%)	2,235 ms:
JavaScript.Function.prototype.call ()	3,085 ms (4.4%)	2,235 ms:
JavaScript.app ()	3,085 ms (4.4%)	2,235 ms:
JavaScript.handle ()	3,085 ms (4.4%)	2,235 ms:
JavaScript.next ()	3,085 ms (4.4%)	2,235 ms:
JavaScript.process_params ()	3,085 ms (4.4%)	2,235 ms:
JavaScript.anonymous ()	3,085 ms (4.4%)	2,235 ms:
JavaScript.trim_prefix ()	3,085 ms (4.4%)	2,235 ms:
JavaScript.handle ()	3,085 ms (4.4%)	2,235 ms:
JavaScript.query ()	3,085 ms (4.4%)	2,235 ms:
JavaScript.next ()	3,085 ms (4.4%)	2,235 ms:
JavaScript.process_params ()	3,085 ms (4.4%)	2,235 ms:
JavaScript.anonymous ()	3,085 ms (4.4%)	2,235 ms:
JavaScript.trim_prefix ()	3,085 ms (4.4%)	2,235 ms:
JavaScript.handle ()	3,085 ms (4.4%)	2,235 ms:
JavaScript.expressInit ()	3,085 ms (4.4%)	2,235 ms:
JavaScript.next ()	3,085 ms (4.4%)	2,235 ms:
JavaScript.process_params ()	3,085 ms (4.4%)	2,235 ms:
JavaScript.anonymous ()	3,085 ms (4.4%)	2,235 ms:
JavaScript.handle ()	3,085 ms (4.4%)	2,235 ms:
JavaScript.dispatch ()	3,085 ms (4.4%)	2,235 ms:
JavaScript.next ()	3,085 ms (4.4%)	2,235 ms:
JavaScript.handle ()	3,085 ms (4.4%)	2,235 ms:
JavaScript.anonymous ()	3,085 ms (4.4%)	2,235 ms:
R.plotcars ()	2,884 ms (4.1%)	2,219 ms:
R.print ()	2,884 ms (4.1%)	2,219 ms:
R.print.trellis ()	2,596 ms (3.7%)	1,963 ms:
R.plot.trellis ()	2,596 ms (3.7%)	1,963 ms:
R.tryCatch ()	896 ms (1.3%)	754 ms:
R.tryCatchList ()	896 ms (1.3%)	754 ms:
R.tryCatchOne ()	896 ms (1.3%)	754 ms:
R.doTryCatch ()	896 ms (1.3%)	754 ms:
R.checkArgsAndCall ()	493 ms (0.7%)	383 ms:



## Polyglot Stack Trace



## Polyglot Heap Dump

<https://www.graalvm.org/docs/reference-manual/compatibility/>

Quickly check if an NPM module, Ruby gem, or R package is compatible with GraalVM.

x CHECK!

## Graal.js

NAME	VERSION	STATUS
json-url	~> 1.0	100.00% tests pass

## Production-Ready

---

Java  
Scala, Groovy, Kotlin  
JavaScript  
Node.js  
Native Image  
VisualVM

## Experimental

Ruby  
R  
LLVM Toolchain

## Visionary

Python  
VSCode Plugin  
GPU Integration  
WebAssembly  
LLVM Backend / Mobile  
Java on Java

Thank you

---

