ORACLE

GraalVM History, Status, Vision

Thomas Wuerthinger
GraalVM Founder & Project Lead
@thomaswue

25-November-2019

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[3] Lab, Purdue University

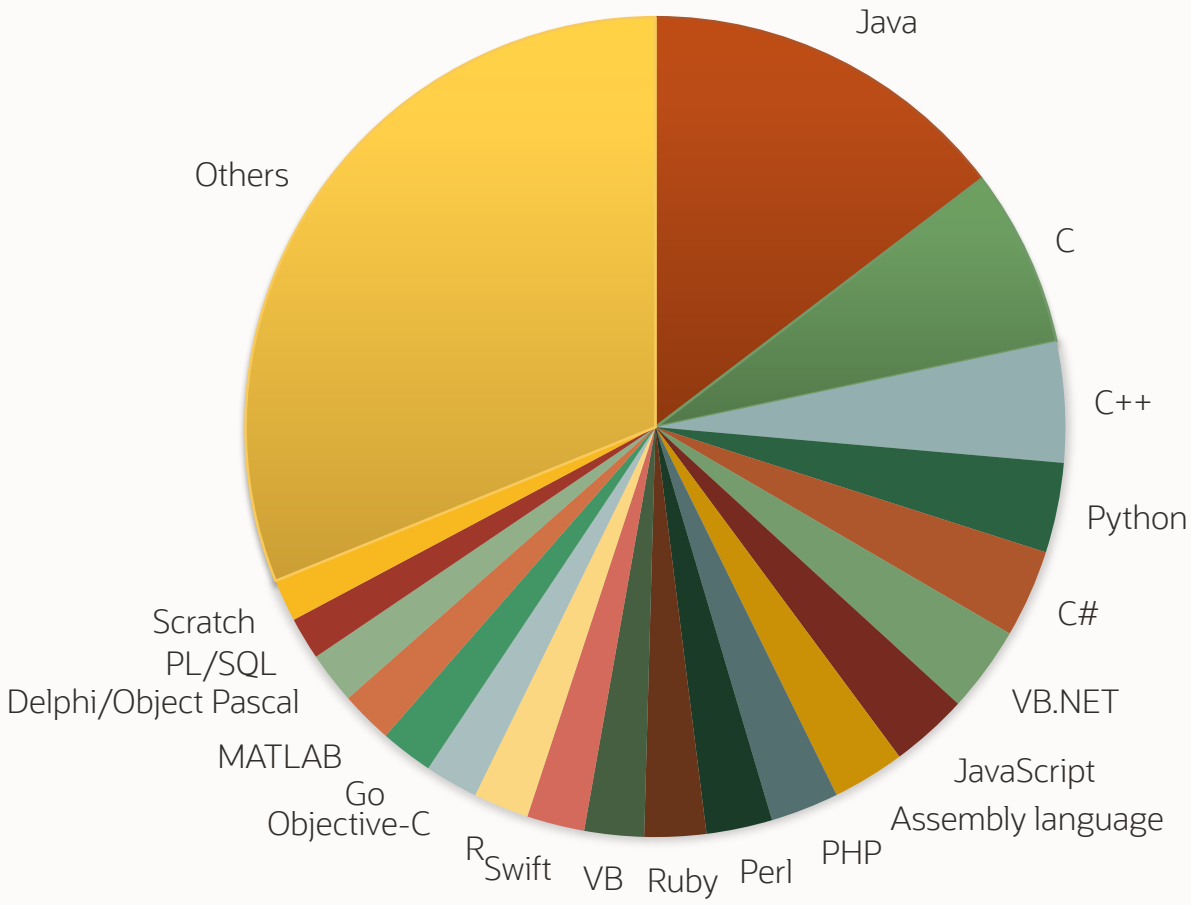
{thomas.wuerthinger, christian.wimmer, doug.simon, mario.wolczko}@oracle.com
{woess, stadler, duboscq, christian.humer}@ssw.jku.at gr@purdue.edu

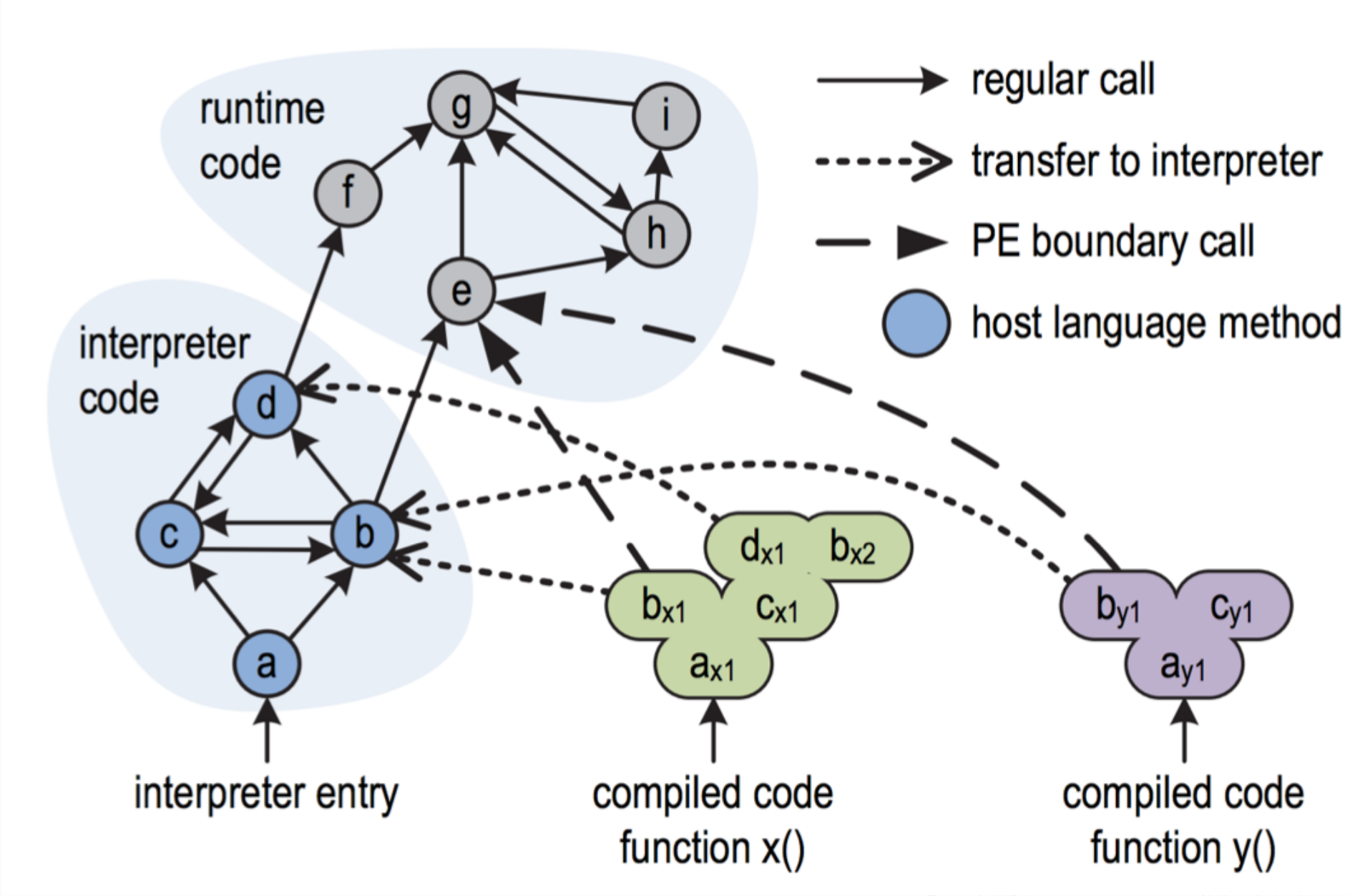
GraalVM will be the Universal VM

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)




runtime
code
interpreter
code
g
i
f
h
e
d
c
b
a
regular call
transfer to interpreter
PE boundary call
host language method
d_{x1} b_{x2}
b_{x1} c_{x1}
a_{x1}
b_{y1} c_{y1}
a_{y1}
interpreter entry
compiled code
function x()
compiled code
function y()

GraalVM™




OpenJDK™

ORACLE®
Database

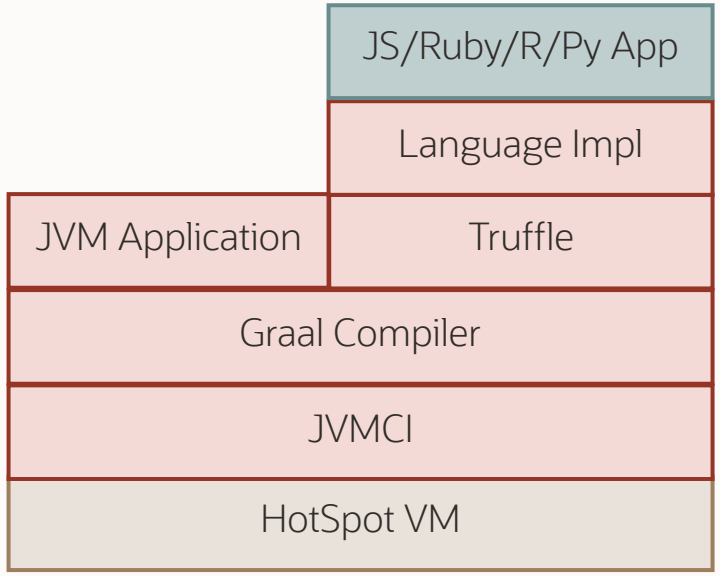
Native Image

What is GraalVM?

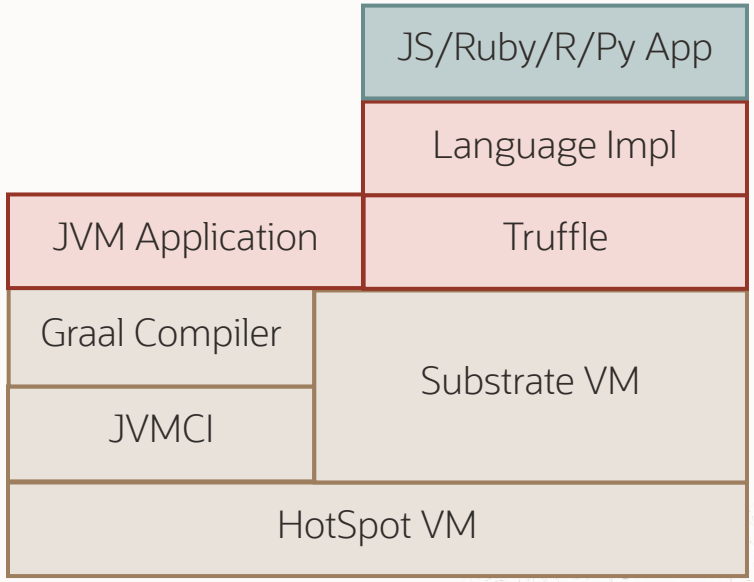
- Drop-in replacement for Java 8 and Java 11
  - Run your Java application faster
- Ahead-of-time compilation for Java
  - Create standalone binaries with low footprint
- High-performance JavaScript, Python, Ruby, R, …
  - The first VM for true polyglot programming
  - Implement your own language or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

Technical Architecture

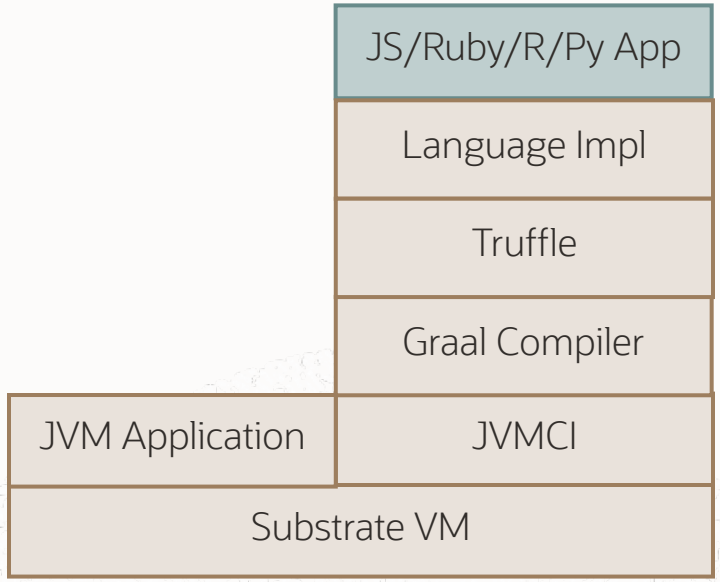
GraalVM JIT „jargraal“



GraalVM JIT „libgraal“

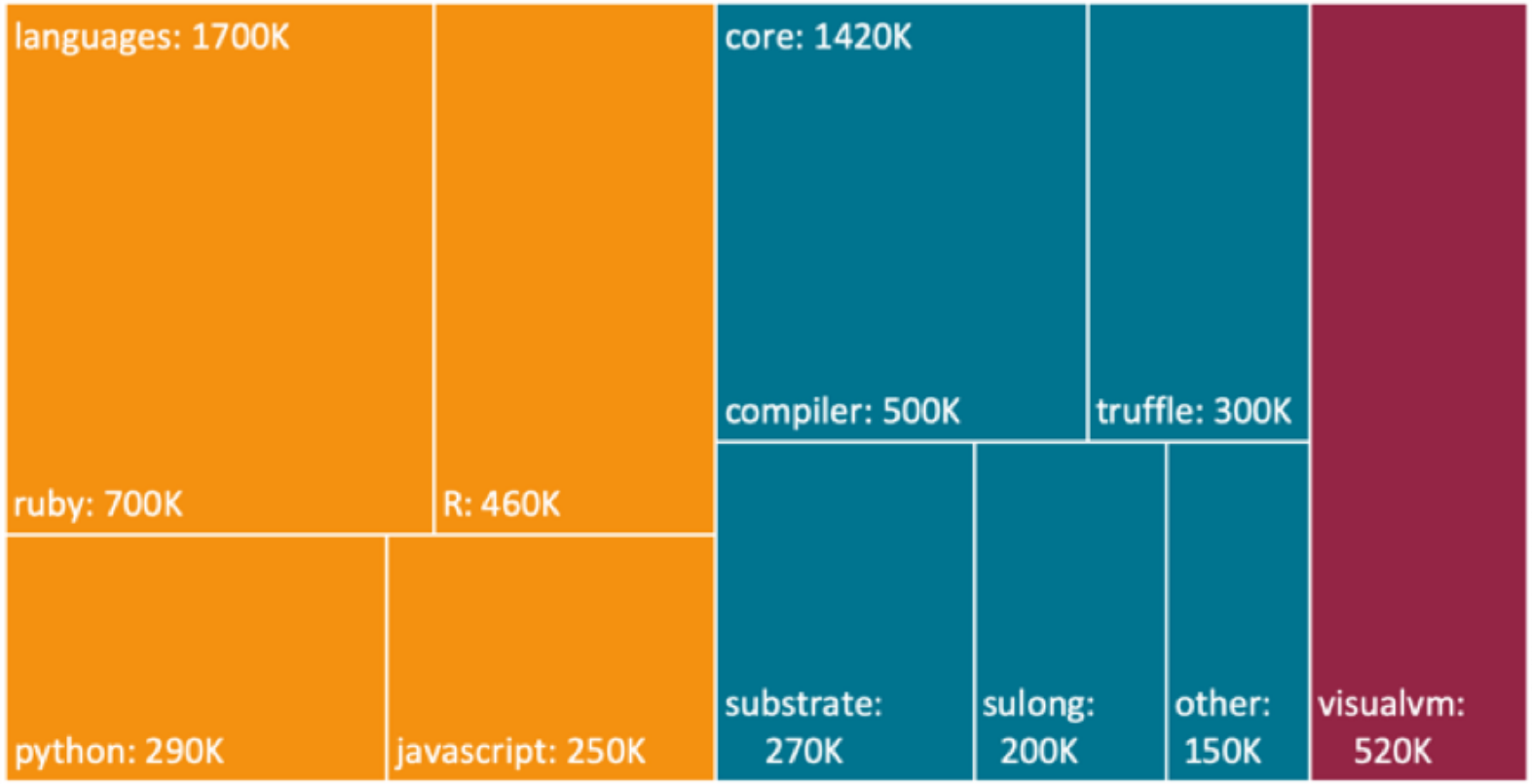


GraalVM AOT (native image)



GraalVM Open Source

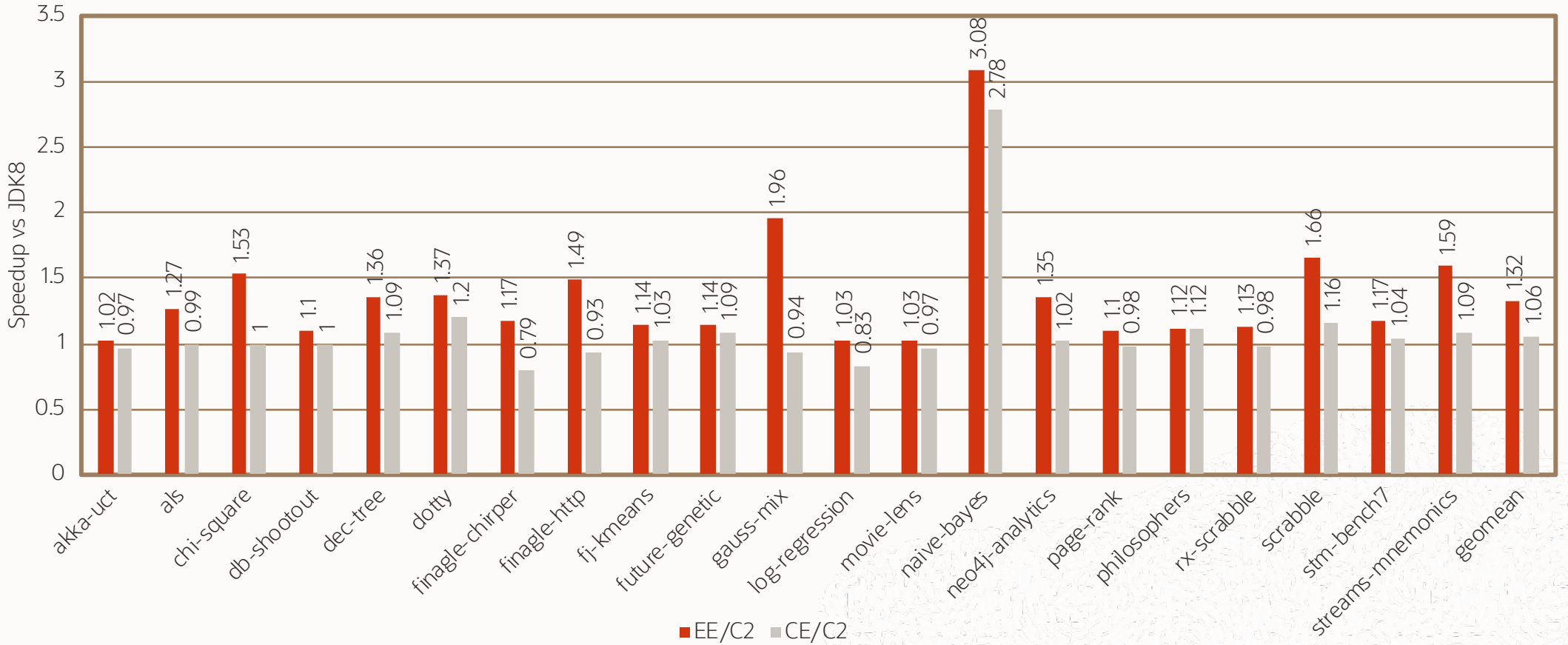
Open Source LOC actively maintained by GraalVM team



Total: 3,640,000 lines of code

GraalVM Vision:
Abstractions should be without performance regret!

GraalVM JIT Performance: Renaissance.dev

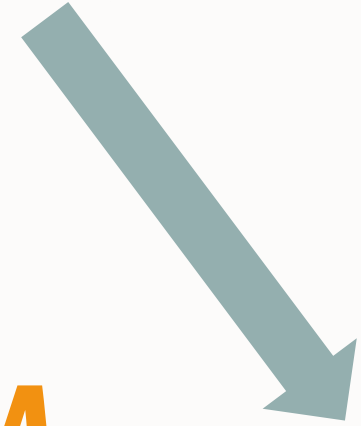


More Benchmarks...

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.



GraalVM

JIT

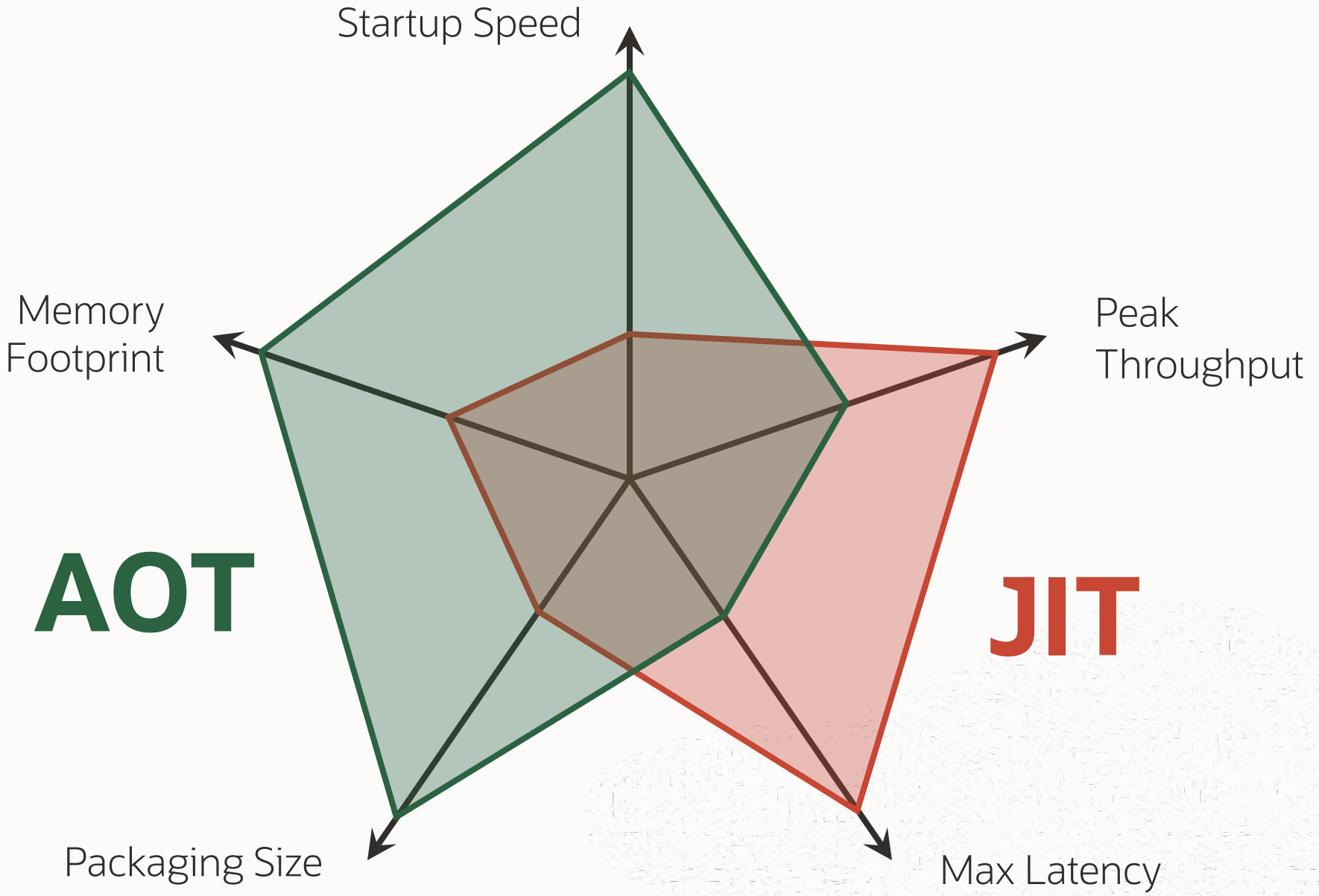
java MyMainClass

OpenJDK

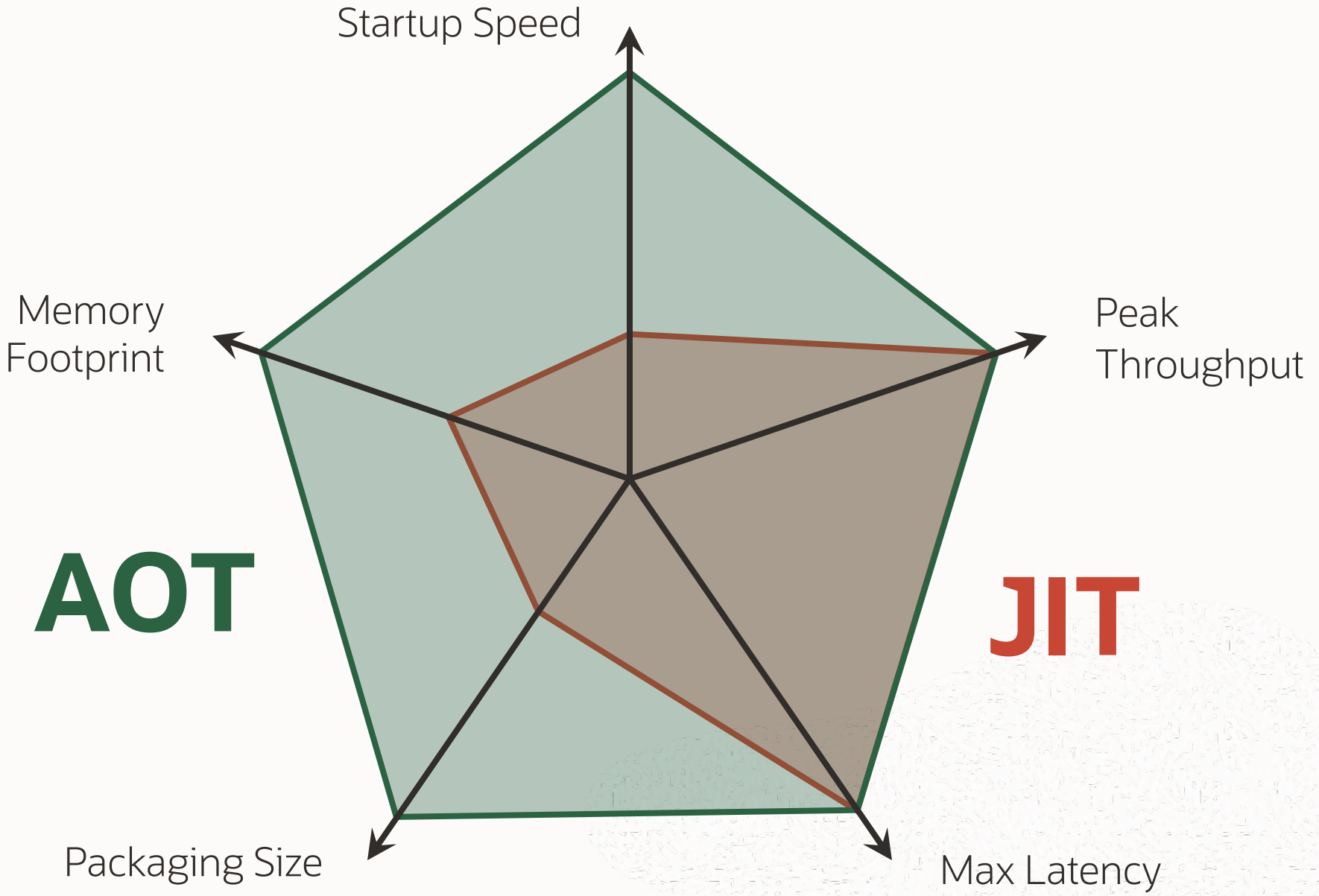
AOT

native-image MyMainClass
./mymainclass

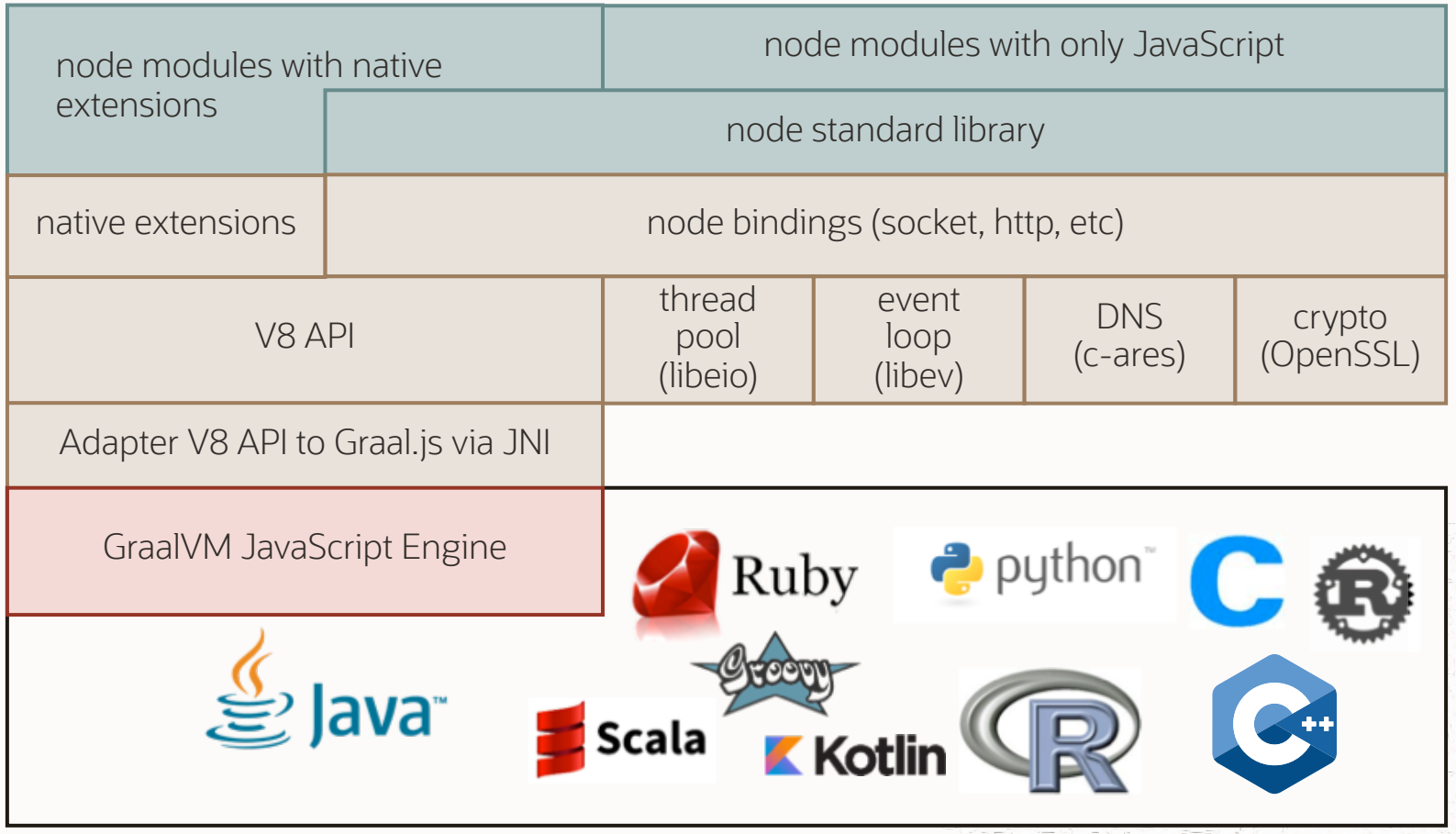
Currently


Startup Speed
Memory Footprint
Peak Throughput
AOT
JIT
Packaging Size
Max Latency

Goal


Startup Speed
Memory Footprint
Peak Throughput
AOT
JIT
Packaging Size
Max Latency

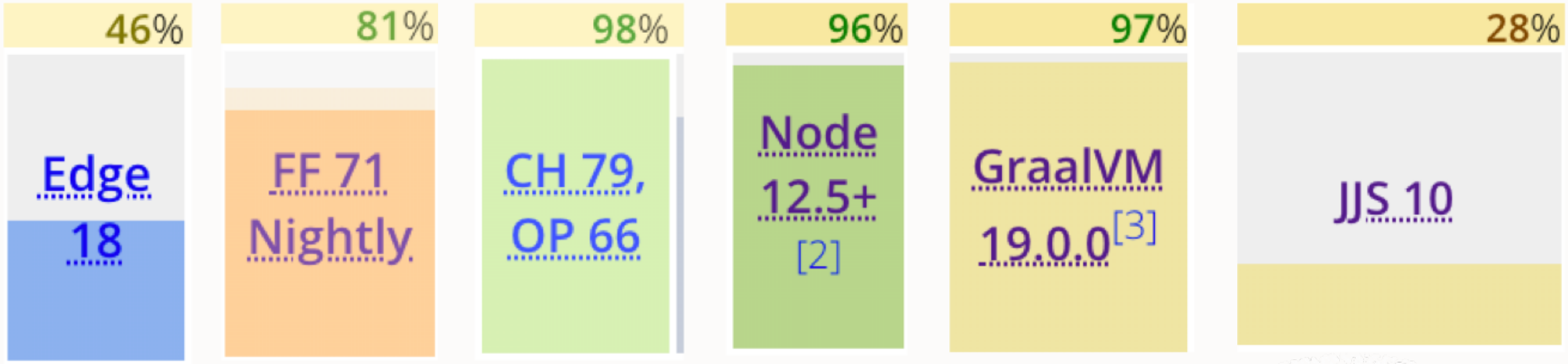
Architecture of Node.js running via GraalVM

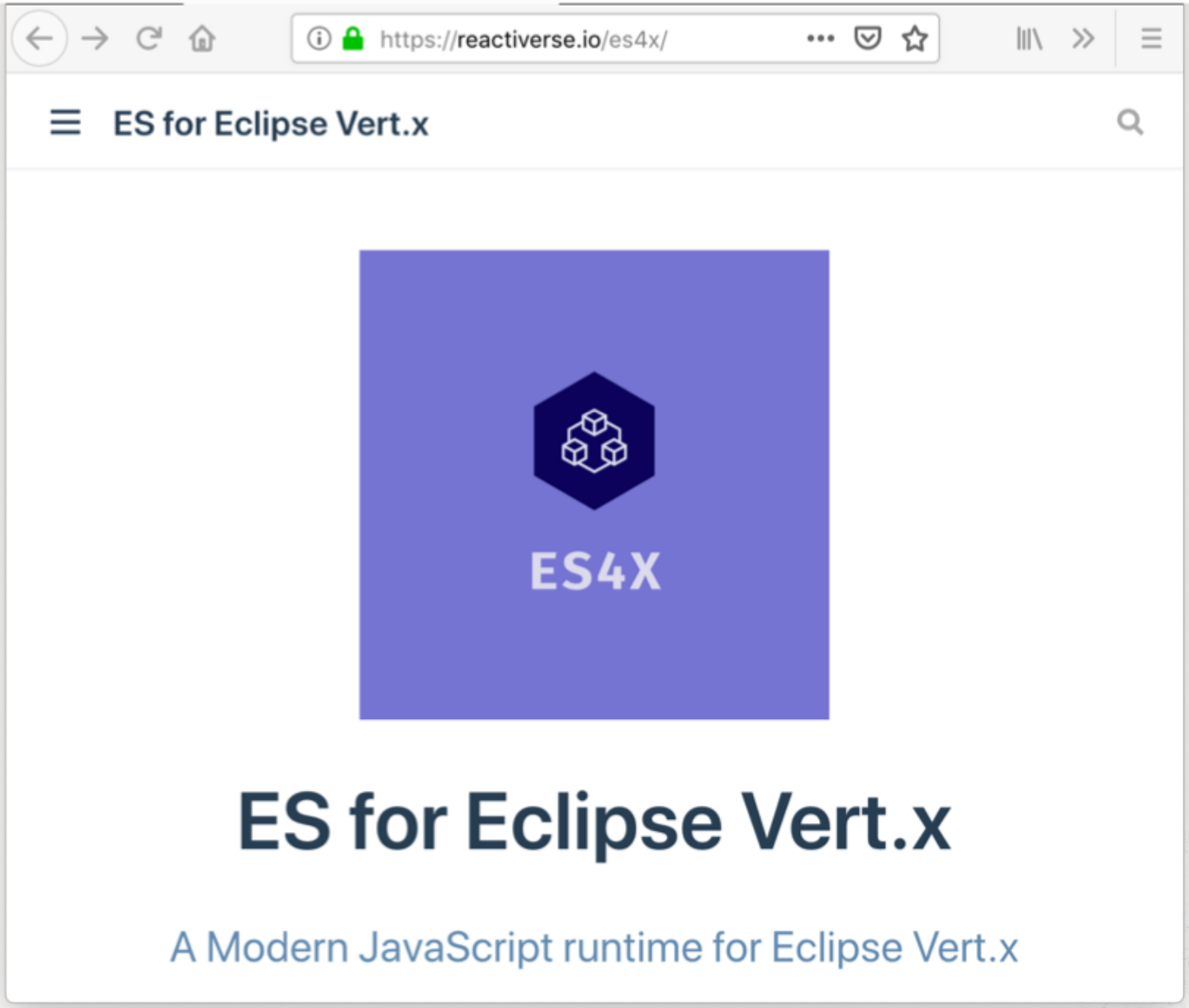


JavaScript

C++

Java


46%
Edge 18
81%
FF 71 Nightly
98%
CH 79, OP 66
96%
Node 12.5+ [2]
97%
GraalVM 19.0.0[3]
28%
JJS 10

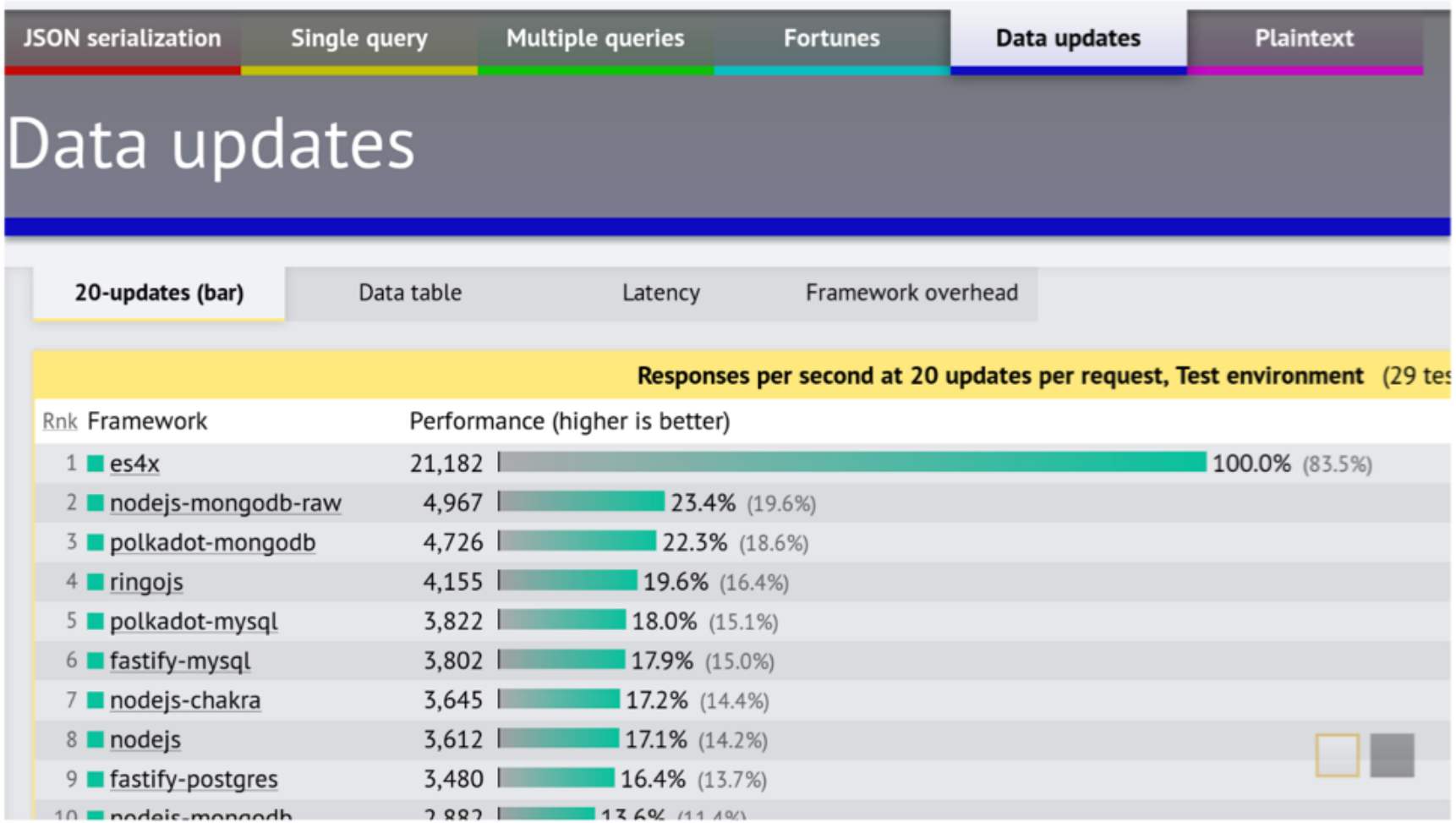


Performant

ES4X runs on top of GraalVM offering a great performance for JavaScript

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:



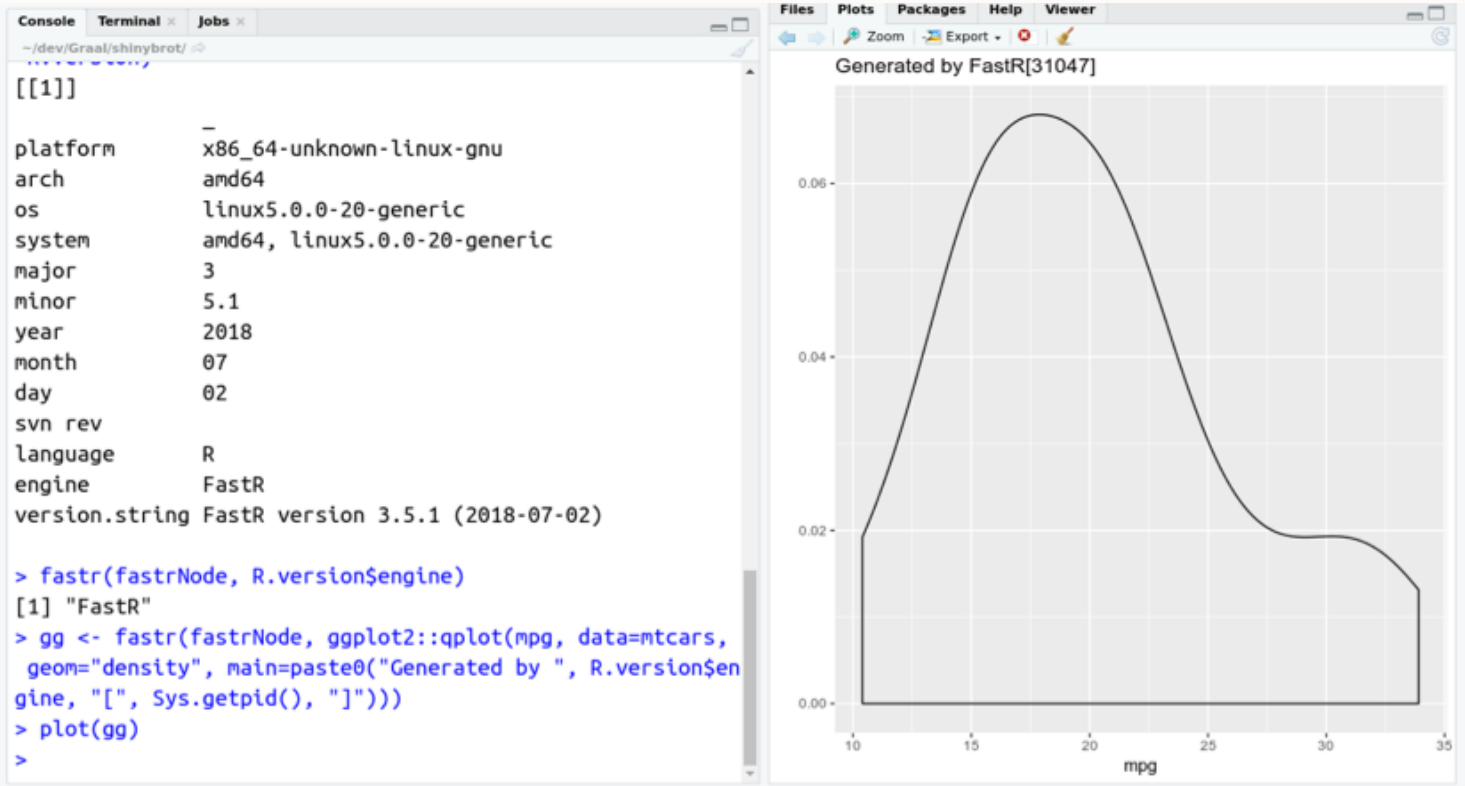
FastR

- 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

FastR Cluster Package

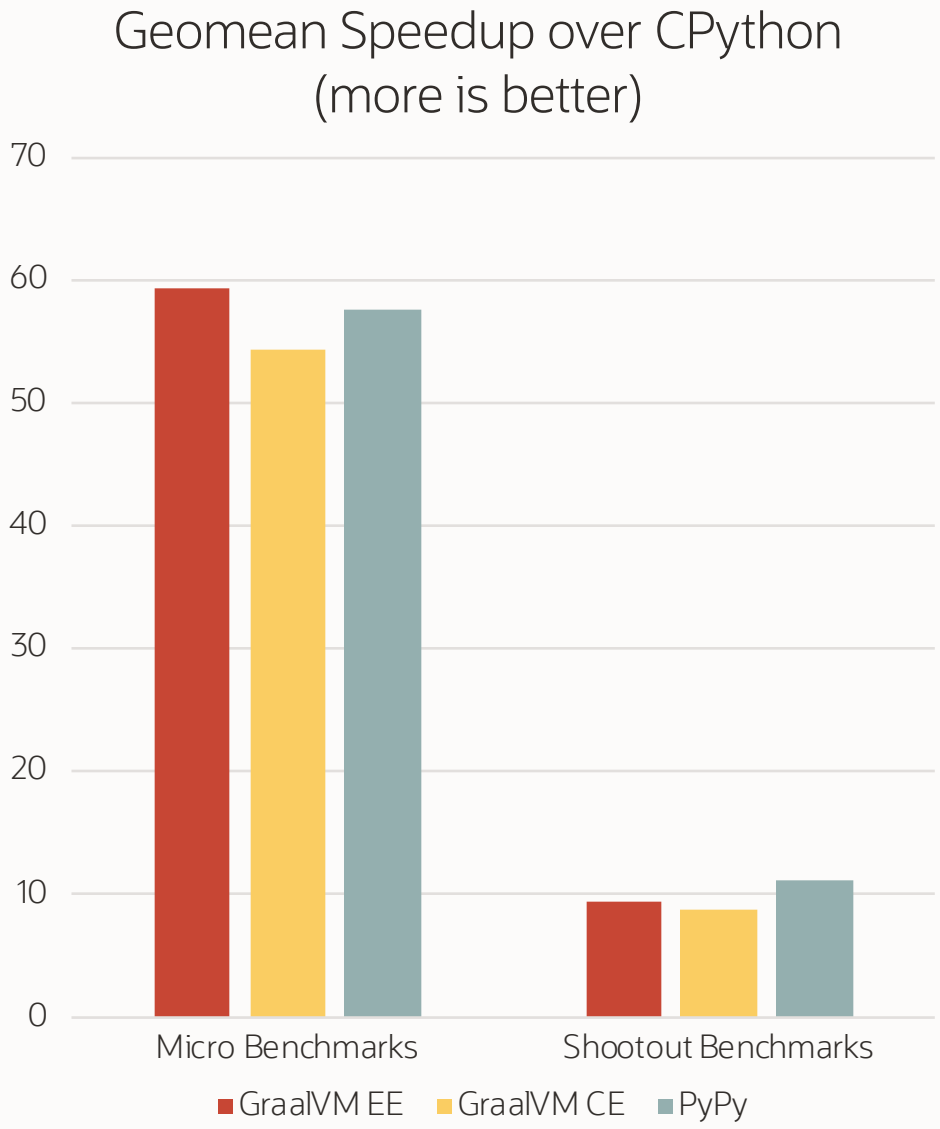
`fastRCluster` package to use FastR as a "cluster" from GNU-R

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



Python Performance

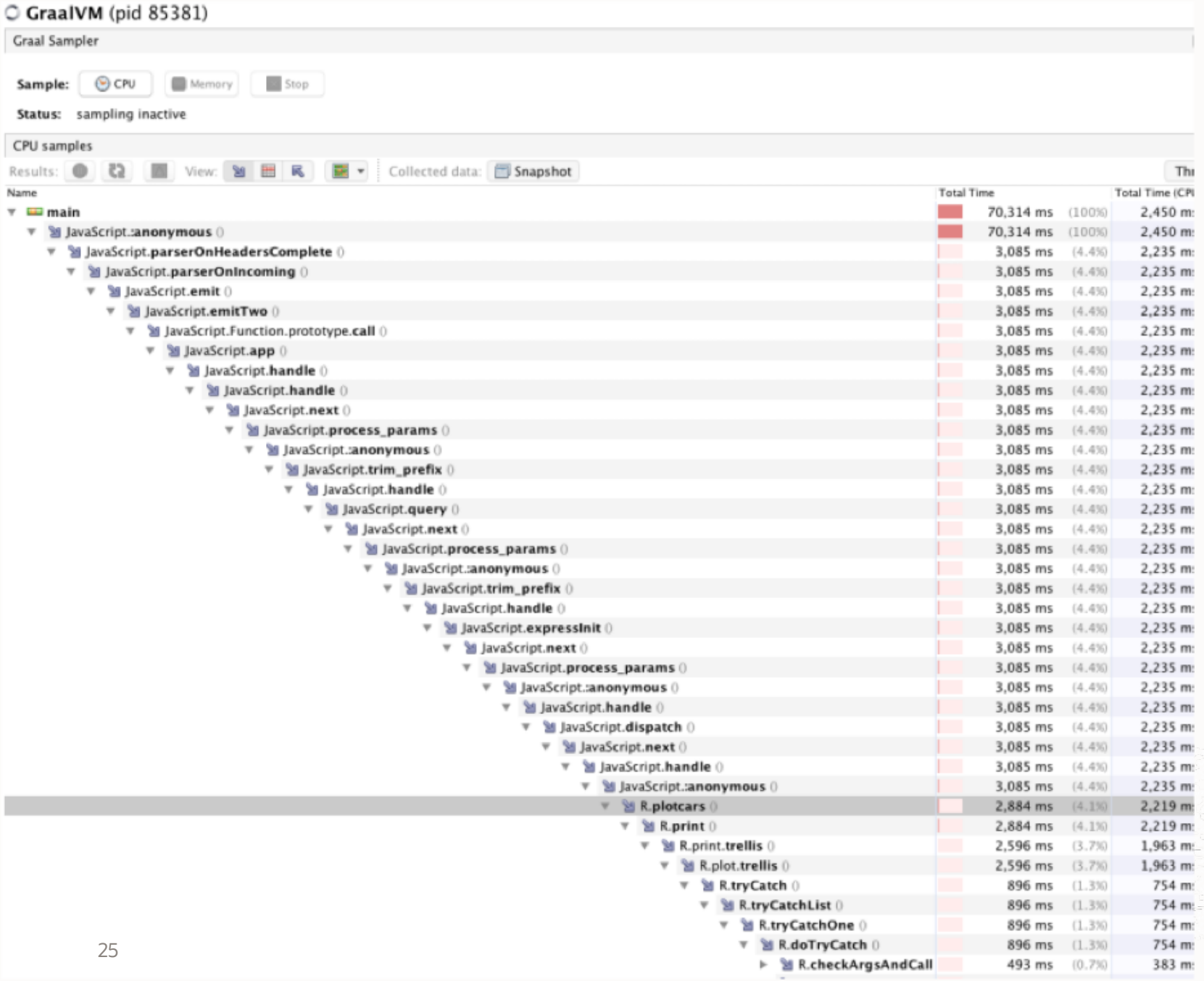
Comparable to PyPy, the fastest alternative



Using Numpy from Java – Calling into Numpy

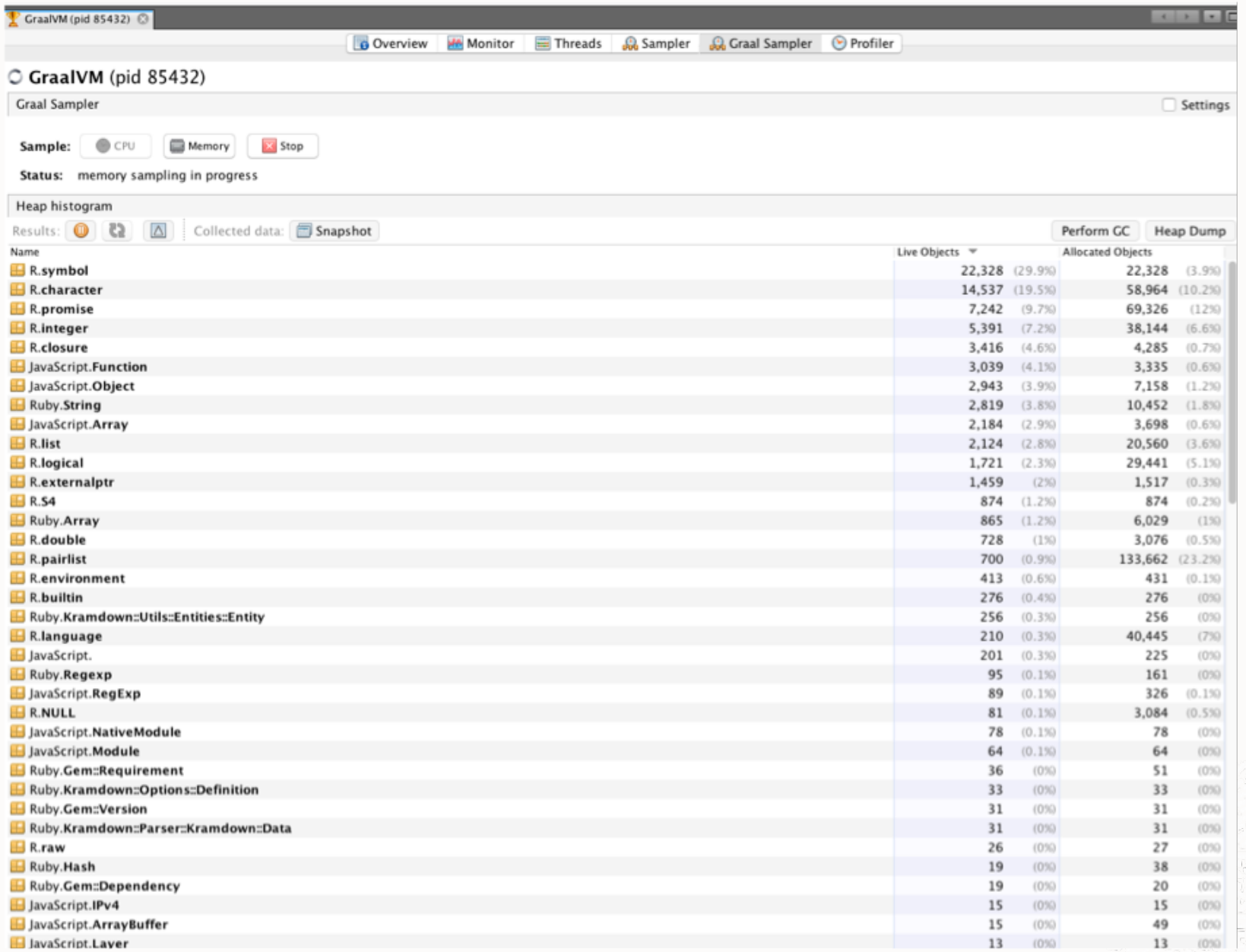
```
try (Context context = Context.newBuilder()
        .option("python.PythonPath", "/path/to/numpy-1.16.4-py3.7-macosx-10.14-x86_64.egg")
        .allowAllAccess(true).build()) {
  Value geomean = context.eval("python", "import numpy\n" + "import math\n" +
      "lambda x: math.pow(numpy.array(x).prod(),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

  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

}
```

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



VisualVM

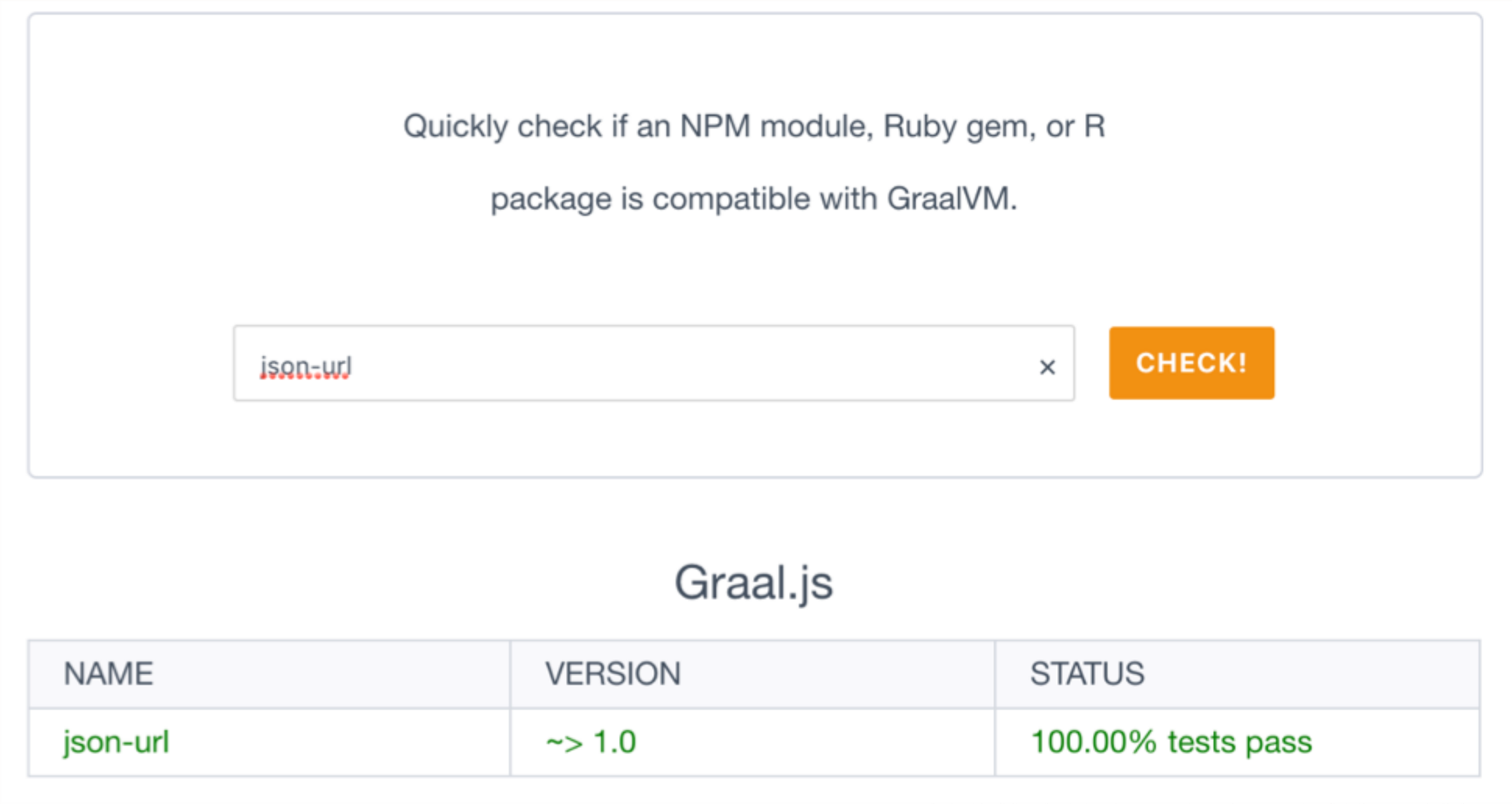
Polyglot Stack Trace



VisualVM

Polyglot Heap Dump

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



Production-Ready	Experimental	Visionary
Java	Ruby	Python
Scala, Groovy, Kotlin	R	VSCode Plugin
JavaScript	LLVM Toolchain	GPU Integration
Node.js		Webassembly
Native Image		LLVM Backend / Mobile
VisualVM		Java on Java

Thank you