JavaOne™

ORACLE®

# Polyglot on the JVM with Graal

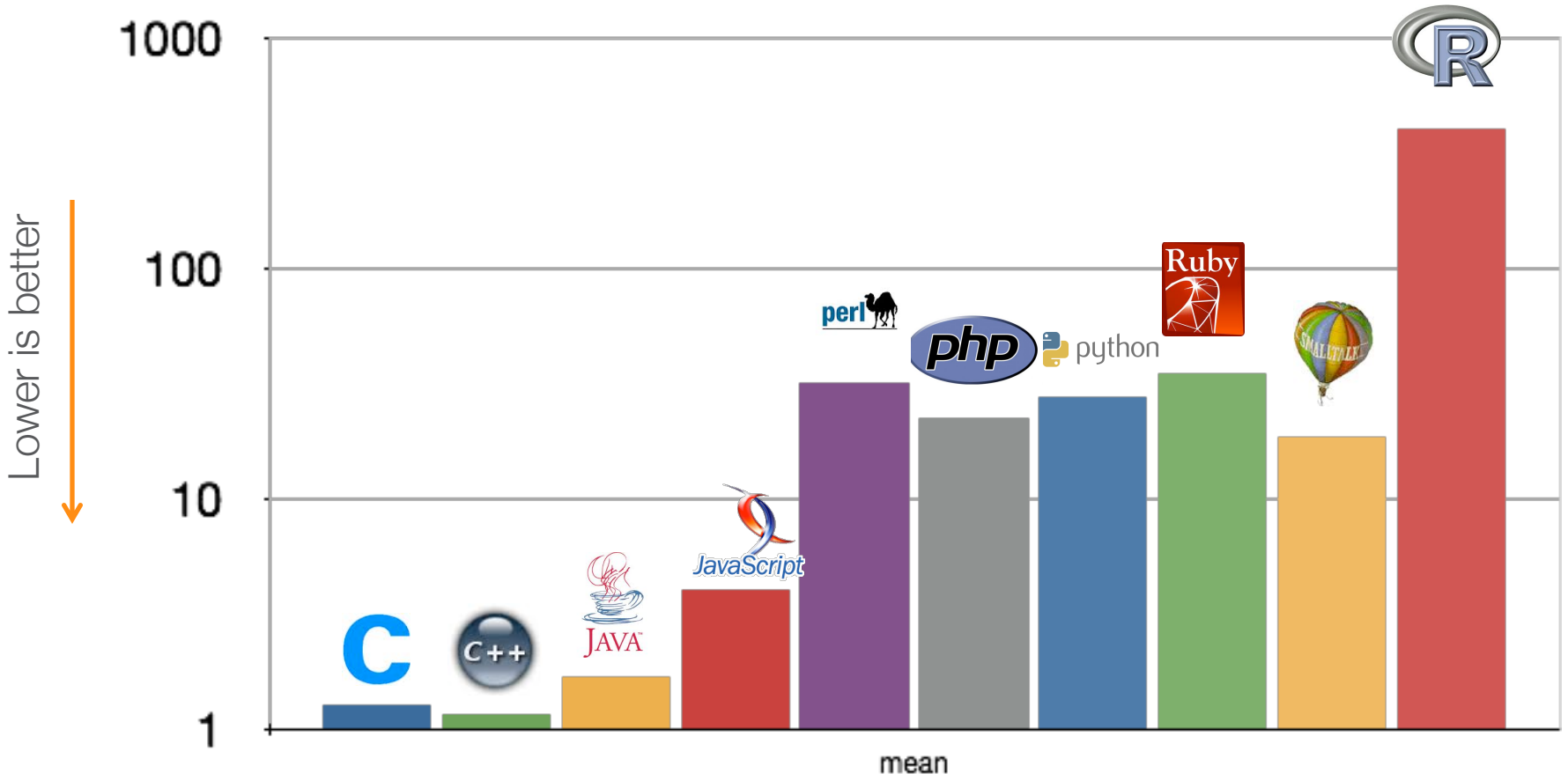
Thomas Wuerthinger
Research Director
Oracle Labs
@thomaswue

September 20, 2016


# Safe Harbor Statement

The following is intended to outline our general product direction. 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 for Oracle's products remains at the sole discretion of Oracle.

# The World Is Polyglot



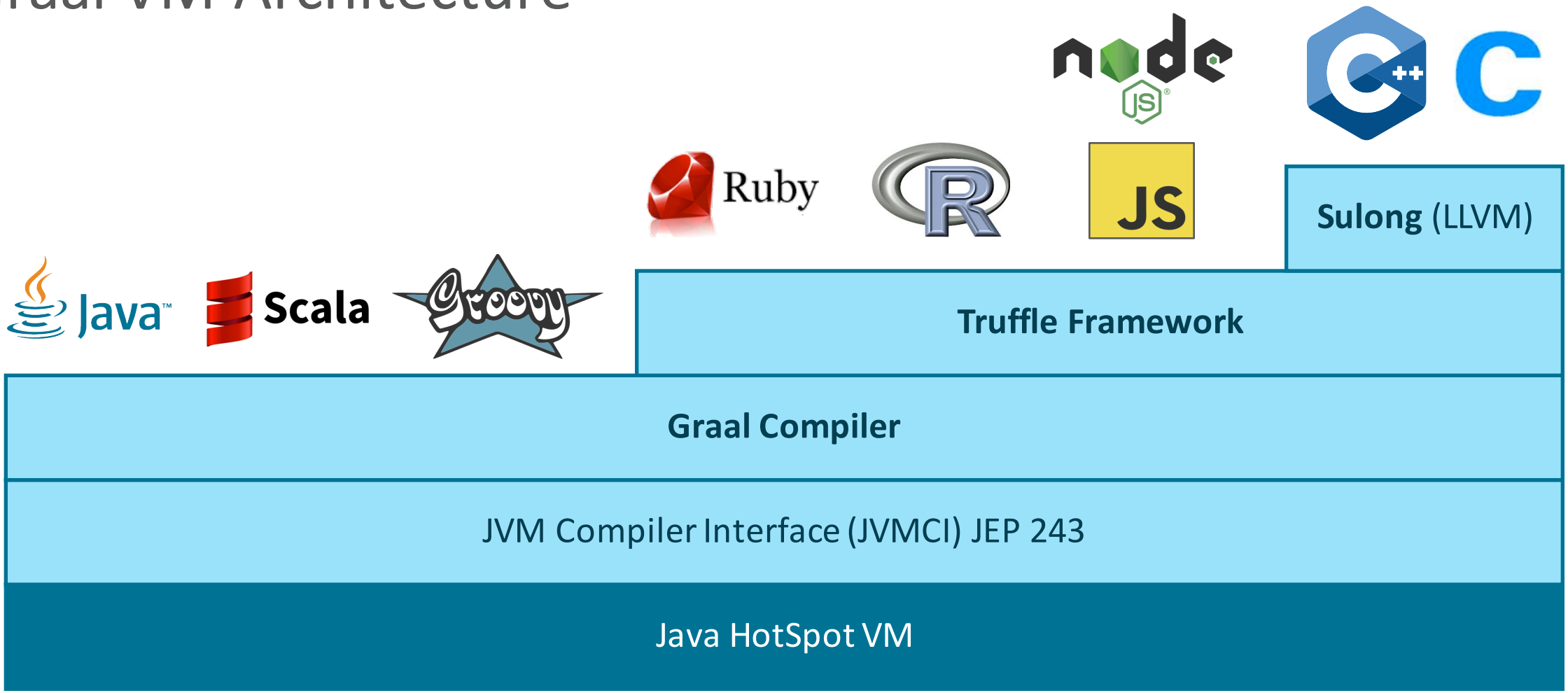
# Graal Vision

**High performance** for all languages

**Zero overhead interoperability** between languages

**Shared infrastructure and tooling** across languages

# Graal VM Architecture



# Graal Compiler Optimizations

inlining, global value numbering, constant folding and propagation, dead code elimination, partial escape analysis, conditional elimination, loop-invariant code motion, core library intrinsifications, invariant reassociation, bounds-checking elimination, read elimination, checkcast elimination, string builder optimizations, test reordering, strength reduction, null check elimination, allocation site merging, speculative guard movement, deoptimization grouping, common subexpression elimination, profile-based devirtualization, class hierarchy analysis, redundant lock elision, tail duplication, path duplication, push-through-phi, de-duplication, alias classification and pointer analysis, induction variable analysis, loop fusion/inversion/unrolling/splitting/unswitching, automatic vectorization, register allocation, instruction selection, peephole optimizations, instruction scheduling, code-block reordering

# Partial Escape Analysis (1)

```
public static Car getCached(int hp, String name) {
    Car car = new Car(hp, name, null);
    Car cacheEntry = null;
    for (int i = 0; i < cache.length; i++) {
        if (car.hp == cache[i].hp &&
              car.name == cache[i].name) {
            cacheEntry = cache[i];
            break;
        }
    }
    if (cacheEntry != null) {
        return cacheEntry;
    } else {

        addToCache(car);
        return car;
    }
}
```

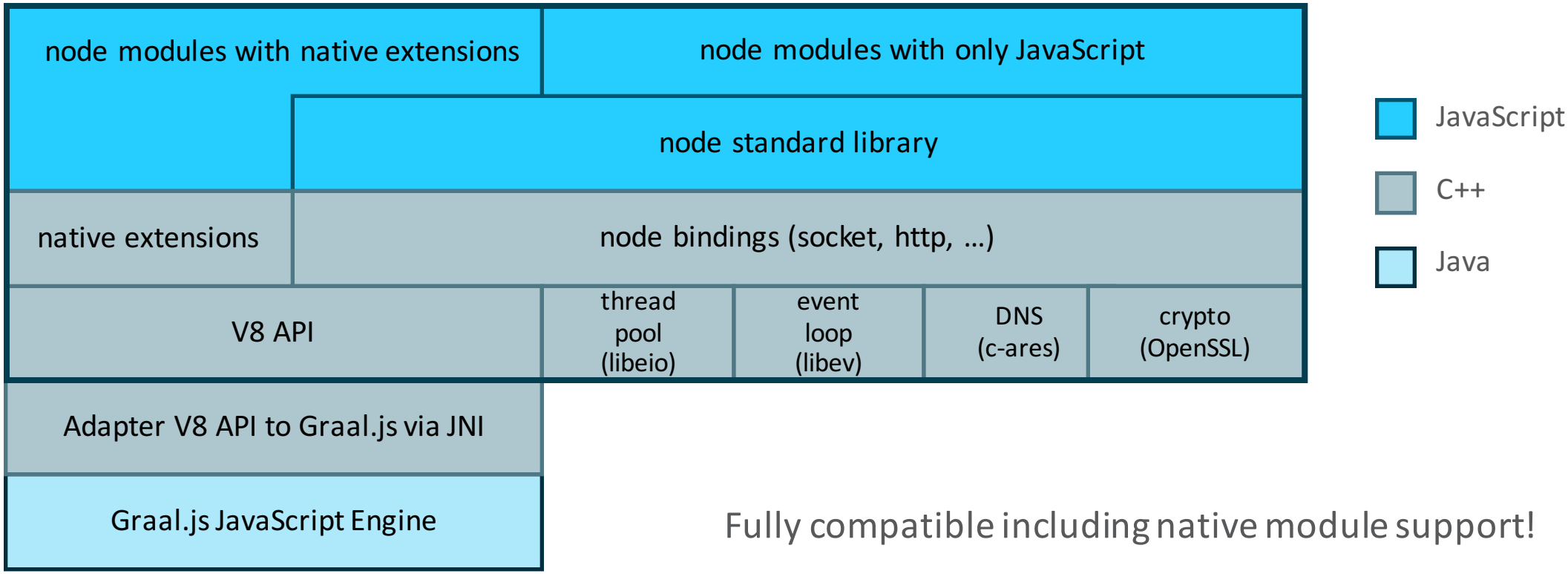
# Partial Escape Analysis (2)

```
public static Car getCached(int hp, String name) {

    Car cacheEntry = null;
    for (int i = 0; i < cache.length; i++) {
        if (hp == cache[i].hp &&
              name == cache[i].name) {
            cacheEntry = cache[i];
            break;
        }
    }
    if (cacheEntry != null) {
        return cacheEntry;
    } else {
        Car car = new Car(hp, name, null);
        addToCache(car);
        return car;
    }
}
```

- `new Car(...)` escapes at:
  - `addToCache(car);`
  - `return car;`
- Might be a very unlikely path
- No allocation in frequent path

# Graal.js Architecture




Fully compatible including native module support!

# Graal OTN Download

- oracle.com/technetwork/oracle-labs/program-languages
- Based on Java 8u92
- Includes a Graal VM technology preview running
  - Java bytecode based languages   
  - JavaScript and node.js  
  - Ruby 
  - R 

# Command Aliases

- graalvm
  - Starts a JVM with Polyglot engine available based on Java 8u92
  - Add –J:-XX:+UseJVMCICompiler for selecting Graal for Java compilations
- js
  - JavaScript engine with full ECMAScript 6 support
- node
  - Running Node.js 6.2.2 with Graal.js replacing V8 as the JavaScript engine
- ruby, irb
  - Running Ruby compatible with version 2.3.1
- R, RScript
  - FastR implementation compatible with GNU R version 3.2.4

# Demo

# Sulong


- Enable LLVM bitcode as just another ”Truffle language”
- Why?
  - Particular interest in running C, C++, and Fortran programs.
  - High-performance native extensions for managed languages.
  - Low overhead of security-related instrumentations such as bounds checks.
  - Apply dynamic optimization techniques to static context.

```
FUNCTION add(x, y)
  INTEGER :: add
  INTEGER :: a
  INTEGER :: b
  add = a + b
  RETURN
END FUNCTION
```

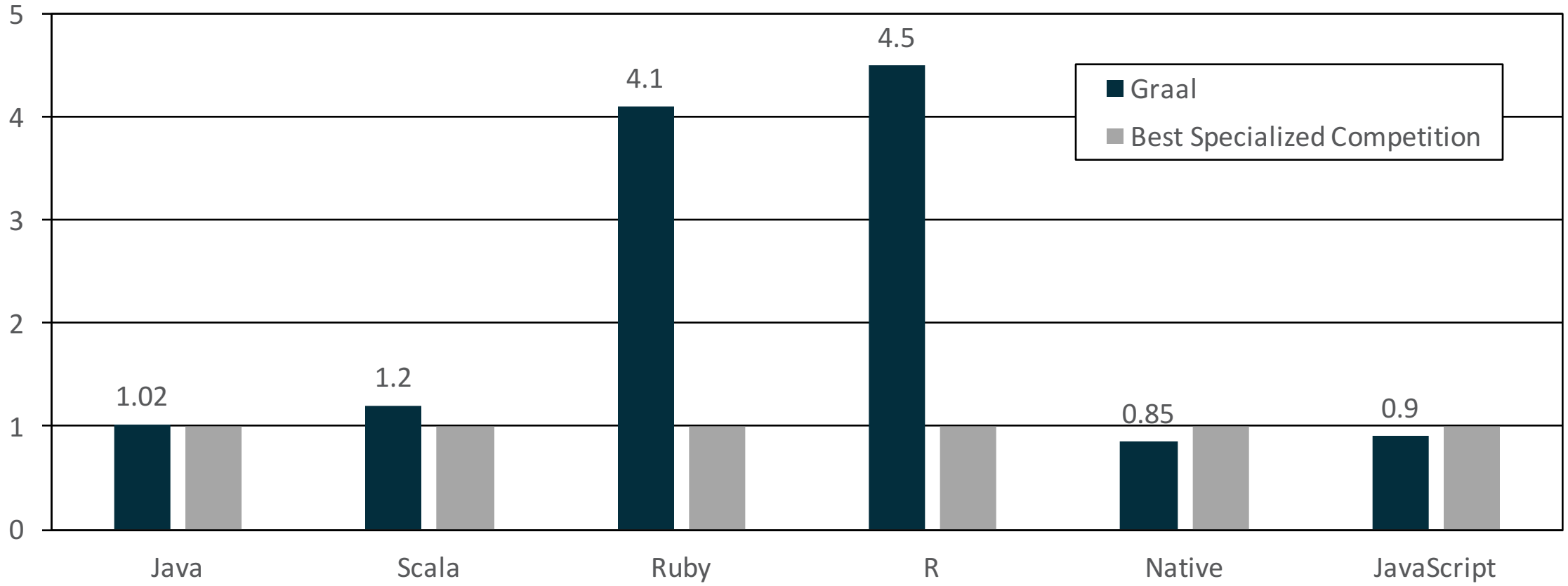
LLVM frontend

```
define i32 @add(i32 %x, i32 %y) #0 {
  %1 = alloca i32, align 4
  %2 = alloca i32, align 4
  store i32 %x, i32* %1, align 4
  store i32 %y, i32* %2, align 4
  %3 = load i32* %1, align 4
  %4 = load i32* %2, align 4
  %5 = add nsw i32 %3, %4
  ret i32 %5
}
```

Graal VM
via Truffle

# Performance Estimates

**Speedup, higher is better**




**Composite performance on 64-bit x86 on well-known benchmark suites relative to recent versions of: HotSpot/Server, JRuby, GNU R, LLVM, V8**

# Open Source

- github.com/graalvm/
- graal-core: dynamic compiler technology
- truffle: language implementation framework
- fastr: implementation of the R runtime
- sulong: execution of LLVM-based languages
- rubytruffle: implementation of the Ruby runtime
- simplelanguage: example language for getting started

# Acknowledgements


**Oracle**
Danilo Ansaloni
Stefan Anzinger
Cosmin Basca
Daniele Bonetta
Matthias Brantner
Petr Chalupa
Jürgen Christ
Laurent Daynès
Gilles Duboscq
Martin Entlicher
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

# Q/A

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

github.com/graalvm

gitter.im/graalvm

**@thomaswue**

# Integrated Cloud

## Applications & Platform Services

JavaOne™

ORACLE®