Polyglot on the JVM with Graal

Thomas Wuerthinger
Senior Research Director, Oracle Labs
@thomaswue

Java User Group Zurich, 15 of December 2016

ORACLE

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.

One language to rule them all?

JavaScript: **One language to rule them all** | VentureBeat

venturebeat.com/.../javascript-**one-language**...
von Peter Yared - in 22 Google+ Kreisen
29.07.2011 - Why code in two different scripting languages, one on the client

Python -- one scripting **language to rule them all**? | Parky's Place
dparkinson.blogspot.com/.../**python-one**-scripting-**la**...
12.12.2012 - Previously I had always put off learning **python** for various reasons, **...**
those other scripting languages and be the **one language to rule them all**.

Q & Stuff: **One Language to Rule Them All** - Java
qstuff.blogspot.com/.../**one-language-to-rule-them-a**...
10.10.2005 - **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** …

One Language to Rule Them All?

Let's ask Stack Overflow…

stackoverflow Questions Tags Tour Users

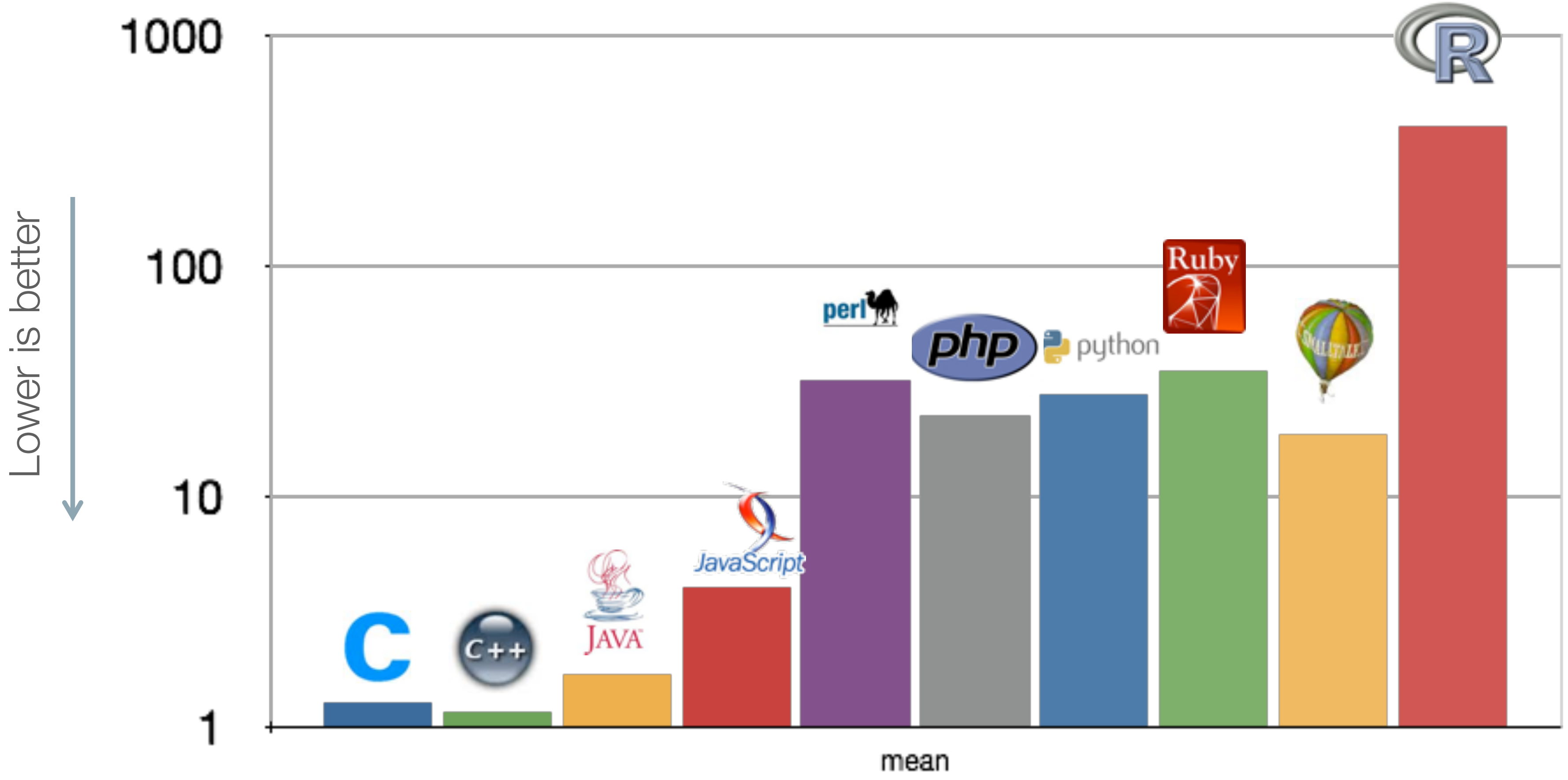
Stack Overflow is a question and answer site for professional and enthusiast programmers. It's 100% free, no registration required.

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

ORACLE

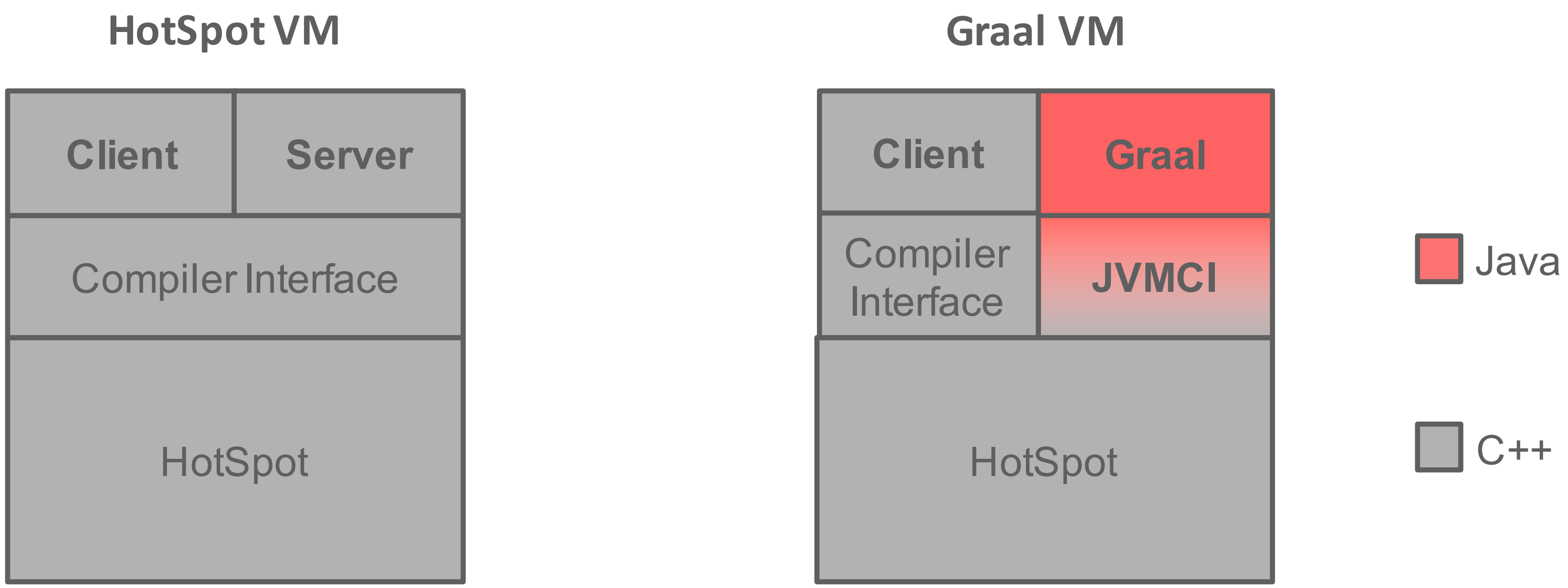
The World Is Polyglot



Graal Overview

A new compiler for HotSpot written in Java and with a focus on speculative optimizations.

JVMCI and Graal included in JDK9, modified version of JDK8 available via OTN.



Compilers Are Complex Beasts…

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

Key Features of Graal

- Designed for speculative optimizations and deoptimization
  - Metadata for deoptimization is propagated through all optimization phases

- Aggressive high-level optimizations
  - Example: partial escape analysis

- Modular architecture
  - Configurable compiler phases

- Written in Java!
  - Easier to maintain and lower entry barrier
  - Blurs the line between user application and user library and compiler
  - Graal compiling and optimizing itself is also a good optimization opportunity
  - https://github.com/graalvm/graal-core

ORACLE

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;
    }
}
```

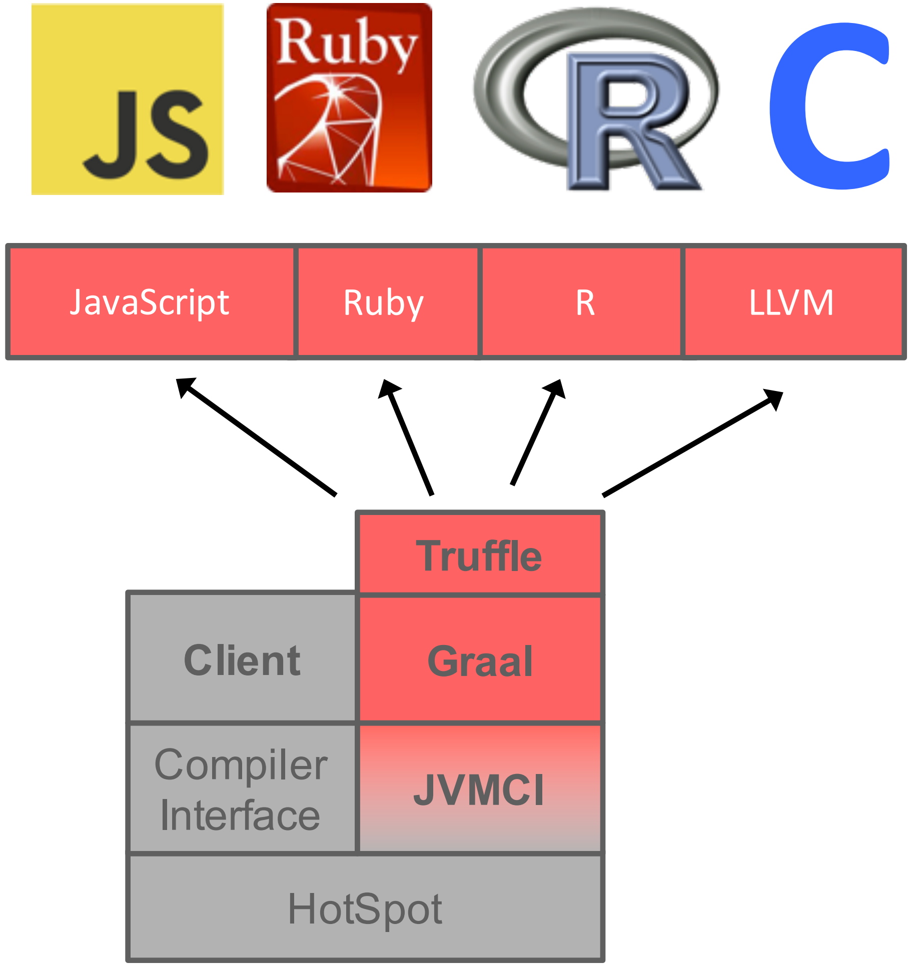

Partial Escape Analysis (2)

```
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 VM Polyglot



- JavaScript
  - Better ECMAScript2016 score than V8
  - Performance competitive with V8
  - Full node.js support
- Ruby
  - Fork of JRuby for ~5-10x speed
- R
  - Statistical language
- C, C++, Fortran
  - Native language support via LLVM

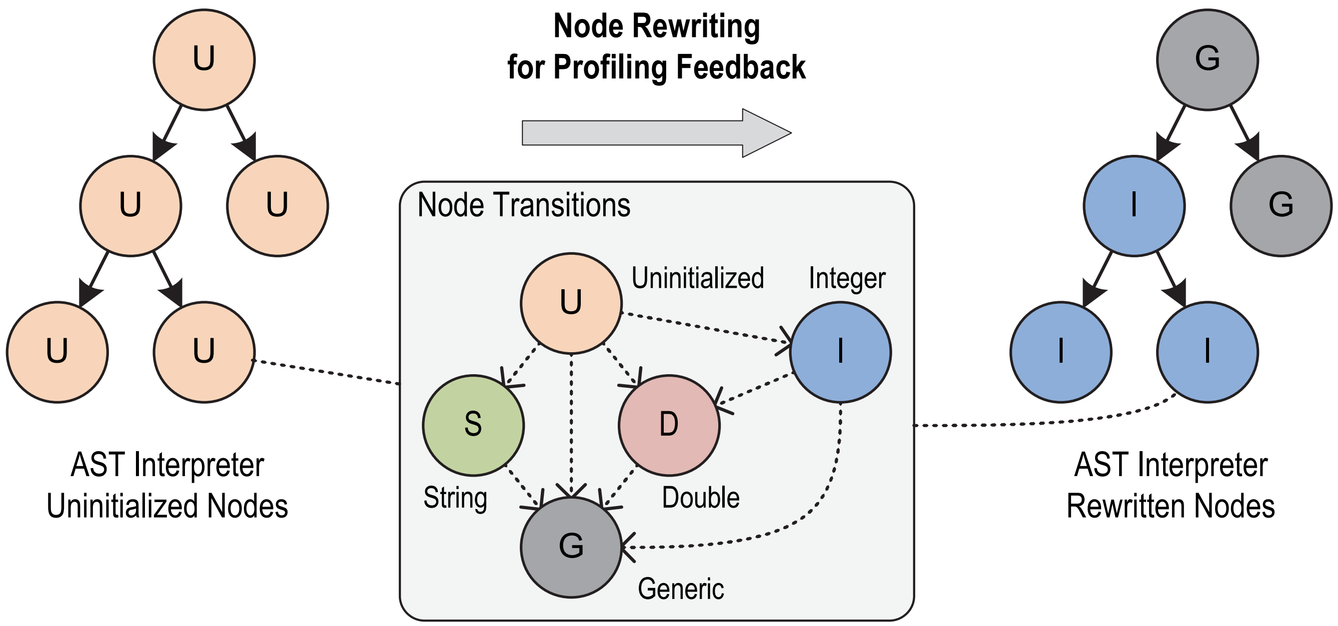
ORACLE

Truffle: System Structure

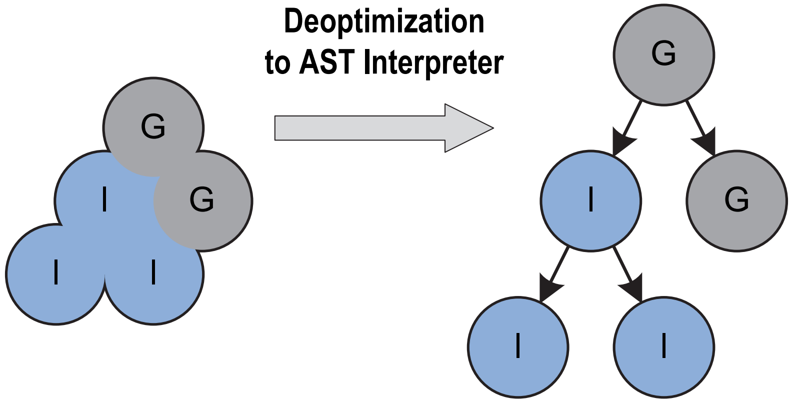
Written by:		Written in:
Application Developer	Guest Language Application	Guest Language
Language Developer	Guest Language Implementation	Managed Host Language
VM Expert	Host Services	Managed Host Language or Unmanaged Language
OS Expert	OS	Unmanaged Language (typically C or C++)

ORACLE

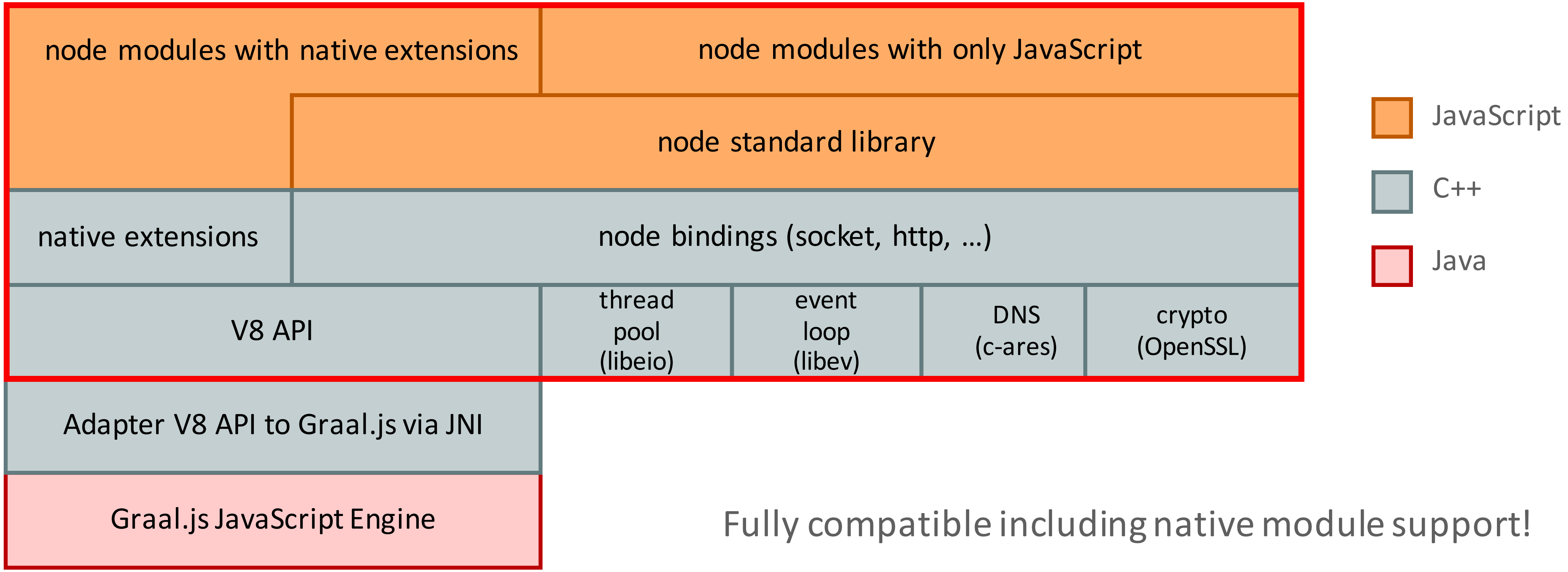
Speculate and Optimize …



… and Deoptimize and Reoptimize!



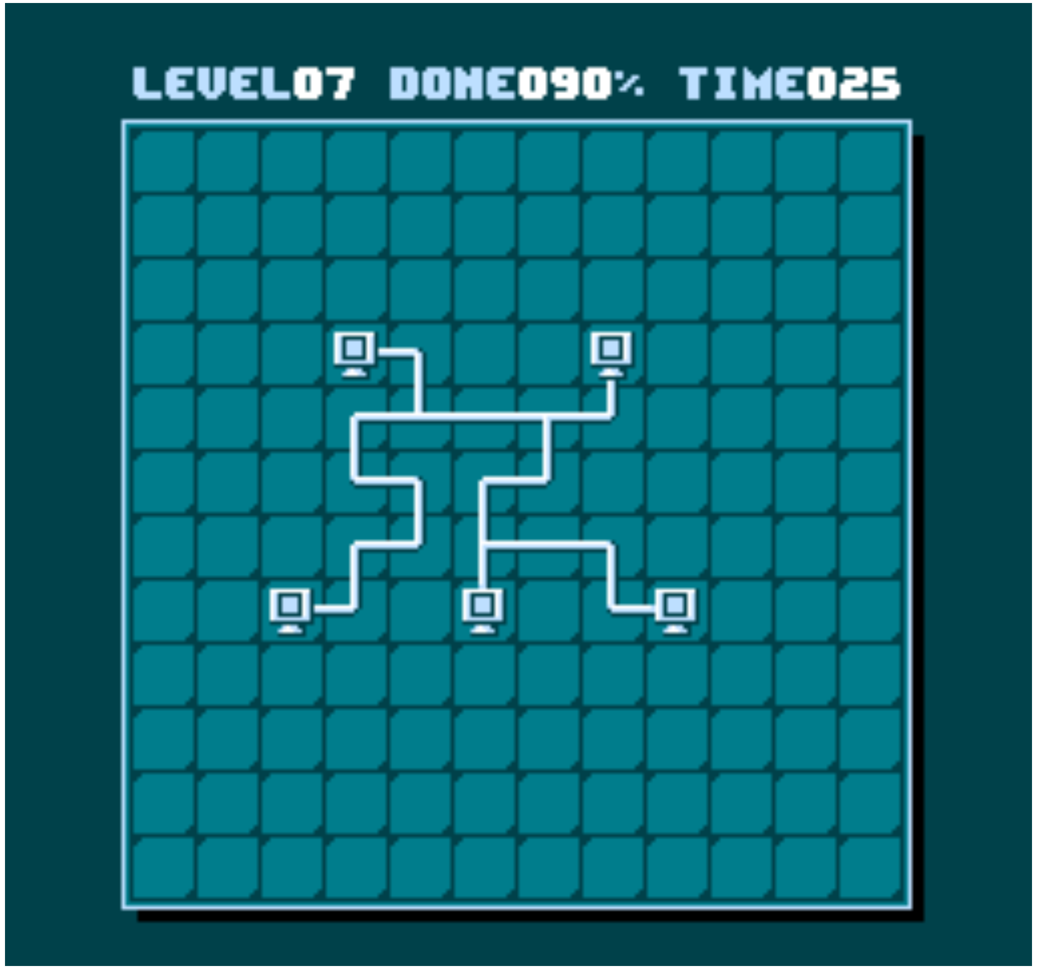
Graal.js Architecture



Fully compatible including native module support!

ORACLE

TruffleRuby – OptCarrot Benchmark



The Ruby team aim to make this NES emulator benchmark 3x faster for their next version, 3.0

It's non-academic code, written based on what the Ruby team thinks is important to optimise

- MRI 2.3.3 runs around ~20 FPS
- JRuby 9.1.6.0 with invokedynamic ~40 FPS
- TruffleRuby on Graal ~180 FPS

https://eregon.me/blog/2016/11/28/optcarrot.html

ORACLE

FastR

https://github.com/graalvm/fastr

- Goal: realize the advantages of the Truffle stack for R
  - Superior performance without resorting to C/C++/Fortran/…
  - Designed for data-heavy and parallel applications
  - CRAN / Bioconductor repository support
- Not an ”incremental improvement” on GNU R
  - New execution engine written from scratch, based on Truffle
  - Designed as a drop-in replacement for GNU R
- Speedup over latest GNU R interpreter
  - Somewhere between 2 and10x

Project Sulong: LLVM front-end for Graal

https://github.com/graalvm/sulong

C/C++

```
int add(x, y) {
  return x + y;
}
```

Fortran

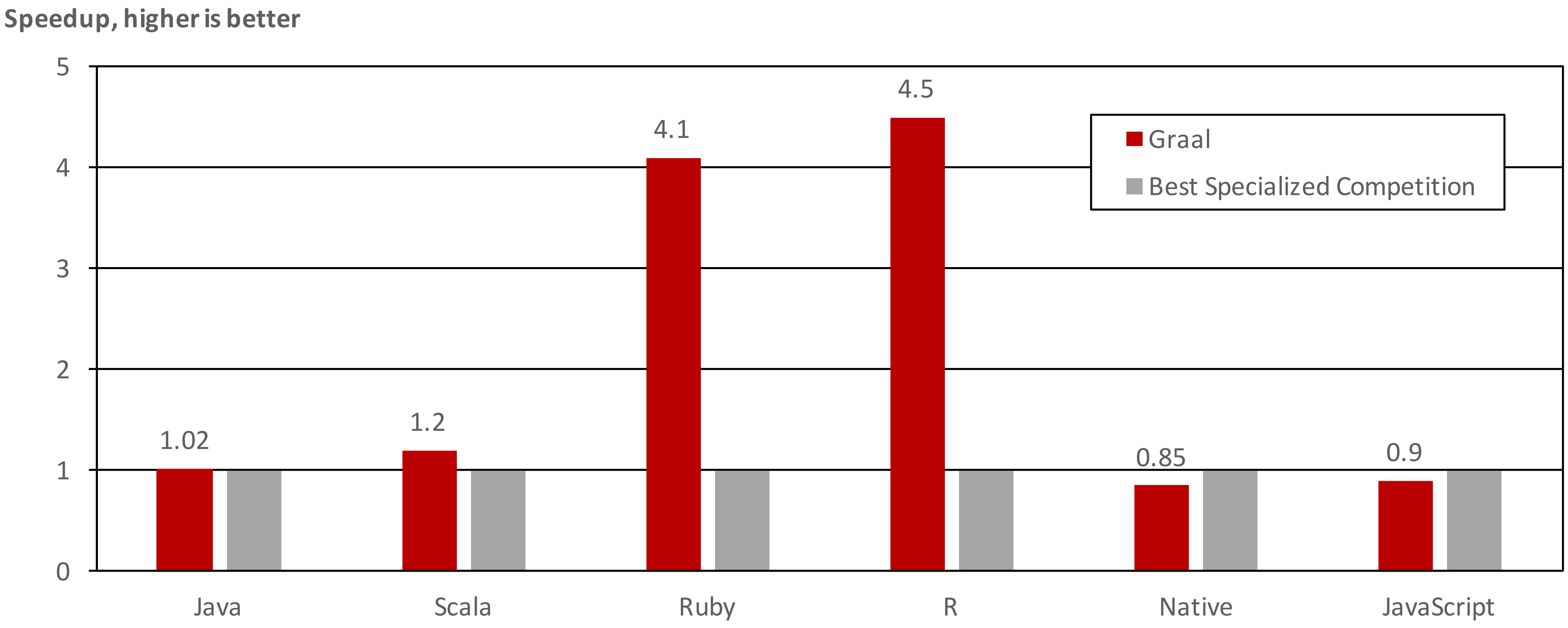
```
FUNCTION add(x, y)
  INTEGER :: add
  INTEGER :: a
  INTEGER :: b
  add = a + b
  RETURN
END FUNCTION
```

Go

```
func add(x int, y int) int {
    return x + y;
}
```

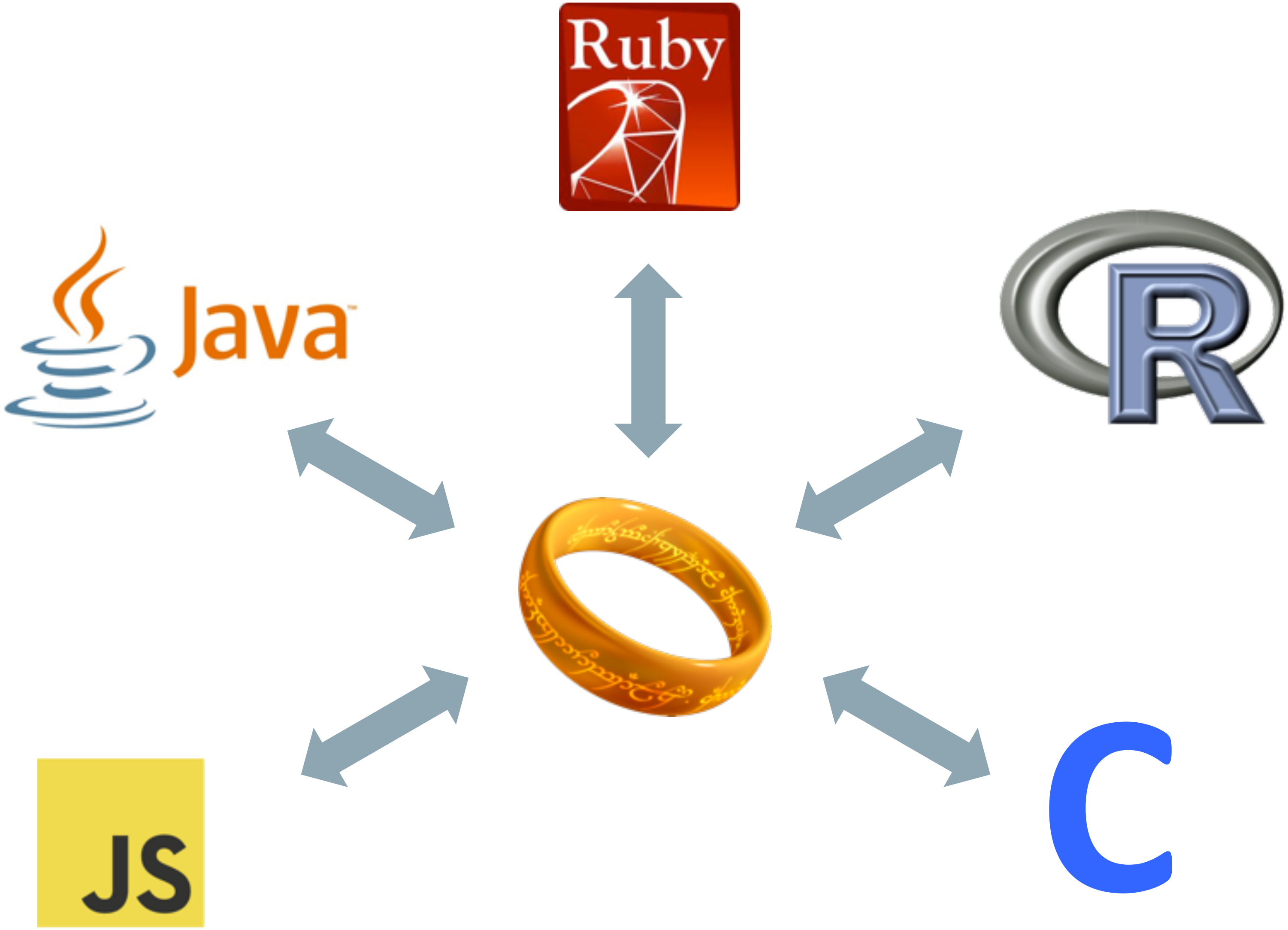
```
define i32 @add(i32 %x, i32 %y) #0 {
  %1 = alloca i32, align 4
  %2 = alloca i32, align 4
  store i32 %x, i32* %1, align 4
  store i32 %y, i32* %2, align 4
  %3 = load i32* %1, align 4
  %4 = load i32* %2, align 4
  %5 = add nsw i32 %3, %4
  ret i32 %5
}
```

Performance: Graal VM



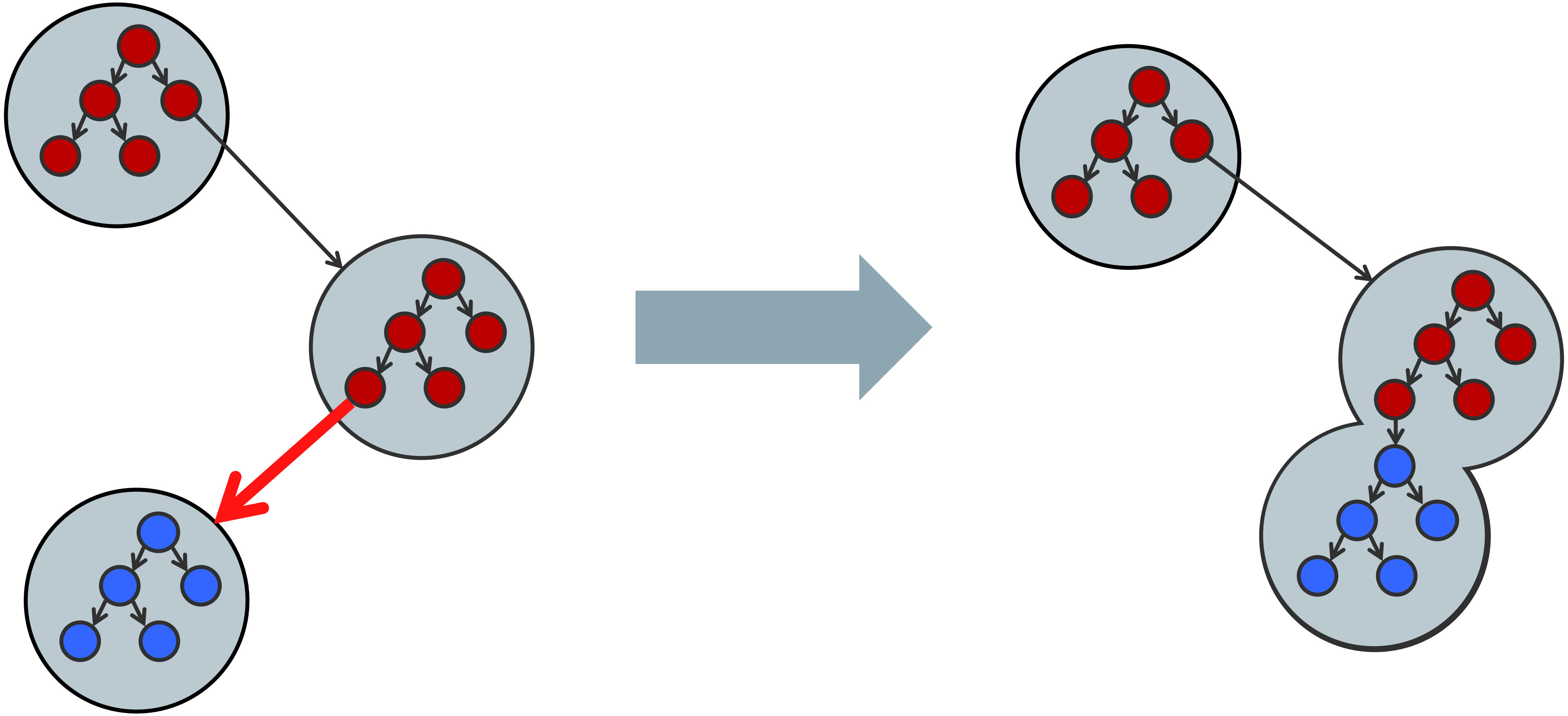
Performance relative to:

HotSpot/Server, HotSpot/Server running JRuby, GNU R, LLVM AOT compiled, V8



- Interoperability
  - Java ⇔ languages
  - Between languages

Inlining Across Language Boundaries



Compilation Across Language Boundaries

Mixed JavaScript and Ruby source code:

```
function main() {
  eval("application/x-ruby",
       "def add(a, b) a + b; end;");
  eval("application/x-ruby",
       "Truffle::Interop.export_method(:add);");
  ...
}

function loop(n) {
  add = import("add");

  i = 0;
  sum = 0;
  while (i <= n) {
    sum = add(sum, i);
    i = add(i, 1);
  }
  return sum;
}
```

Machine code for loop:

```
     mov  r14, 0
     mov  r13, 0
     jmp  L2
L1:  safepoint
     mov  rax, r13
     add  rax, r14
     jo   L3
     inc  r13
     mov  r14, rax
L2:  cmp  r13, rbp
     jle  L1
     ...
L3:  call
transferToInterpreter
```

Substrate VM: Execution Model

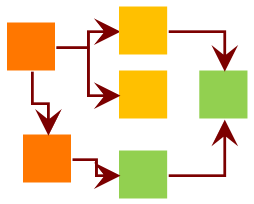
Points-To Analysis

Ahead-of-Time Compilation

Truffle Language

JDK

Substrate VM



Machine Code

Initial Heap

DWARF Info

ELF / MachO Binary

All Java classes from Truffle language (or any application), JDK, and Substrate VM

Reachable methods, fields, and classes

Application running without dependency on JDK and without Java class loading

ORACLE

Open Source

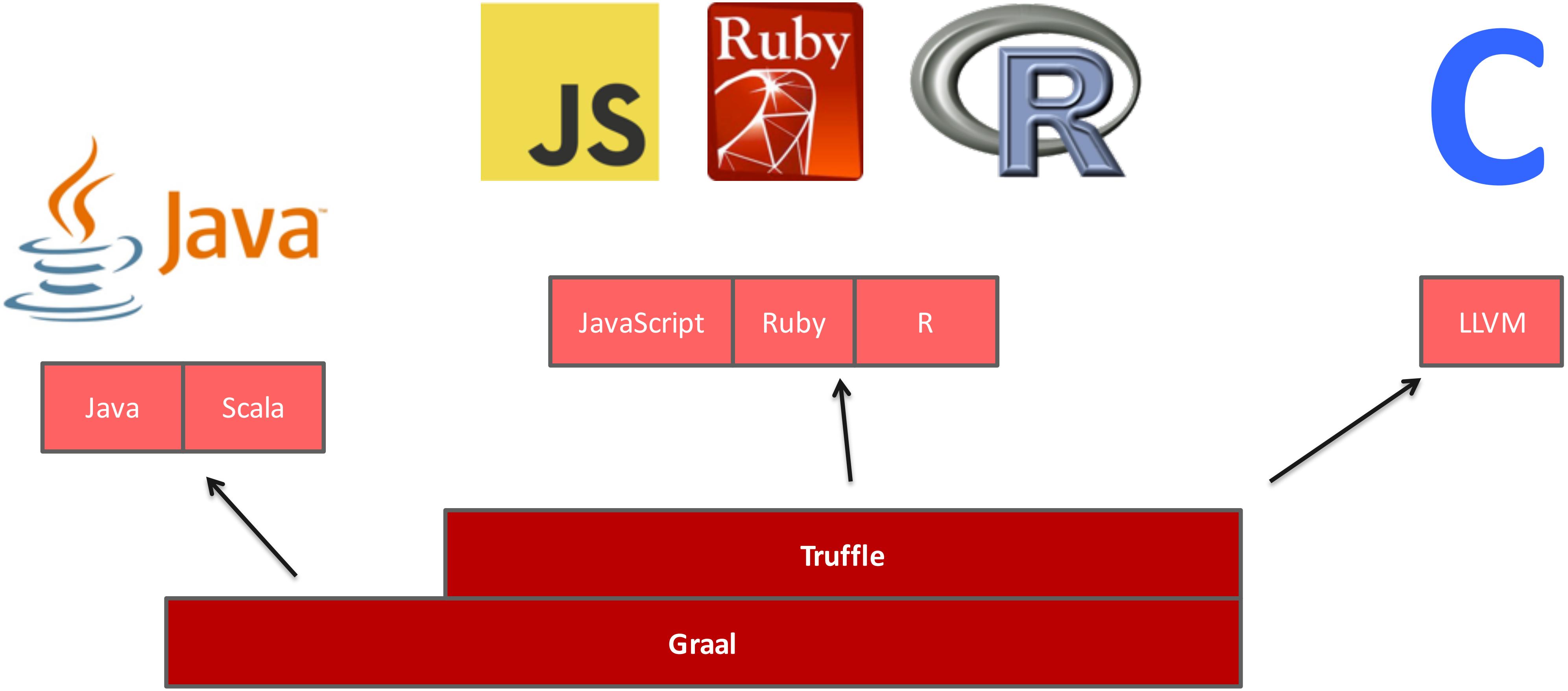
- https://github.com/graalvm/graal-core
  - Graal compiler
- https://github.com/graalvm/truffle
  - Truffle language implementation framework
- https://github.com/graalvm/fastr
  - Fast R runtime
- https://github.com/graalvm/sulong
  - Dynamic runtime for LLVM bitcode
- https://github.com/jruby/jruby/wiki/Truffle
  - Fast Ruby runtime

ORACLE

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

We are looking for example workloads!



@thomaswue

Questions?

Integrated Cloud

Applications & Platform Services

ORACLE