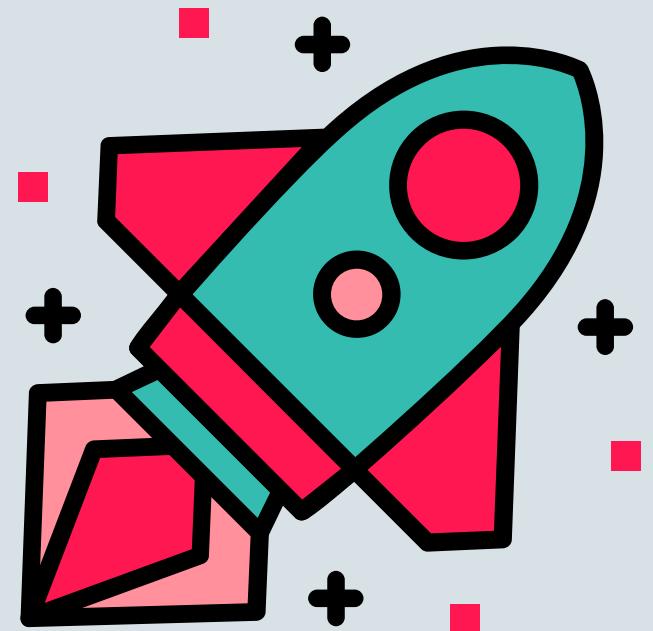


**ORACLE®**

# An Introduction to TruffleRuby

High performance Ruby on GraalVM

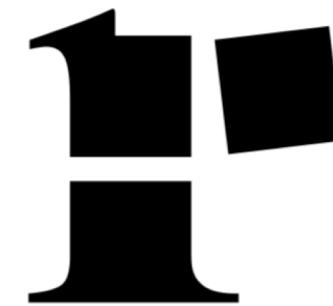
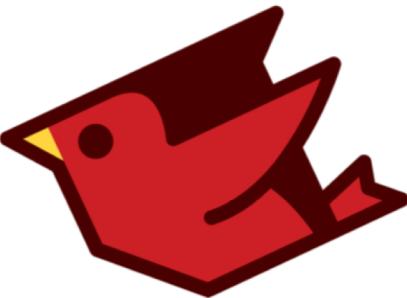
Chris Seaton  
Research Manager  
Oracle Labs  
March 2019



# 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. Oracle reserves the right to alter its development plans and practices at any time, and 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.

# TruffleRuby basics



JRuby logo copyright (c) Tony Price 2011, licensed under the terms of the Creative Commons Attribution-NoDerivs 3.0 Unported (CC BY-ND 3.0)

Ruby logo copyright (c) 2006, Yukihiro Matsumoto, licensed under the terms of the Creative Commons Attribution-ShareAlike 2.5 agreement

Rubinius logo licensed under the terms of the Creative Commons Attribution-NoDerivs 3.0 Unported

Appfolio logo © AppFolio, Inc. 2016

Maglev logo Copyright © 2008-2010 GemStone Systems

OMR logo copyright Eclipse Foundation

# We wanted to build a Ruby that

- Runs idiomatic Ruby code faster
- Runs Ruby code in parallel
- Boots Ruby applications in less time
- Executes C extensions in a managed environment
- Adds fast and low-overhead interoperability with languages like Java, JavaScript, Python and R
- Provides new tooling such as debuggers and monitoring
- All while maintaining very high compatibility with the standard implementation of Ruby

```
$ rbenv install truffleruby-1.0.0-rc14
$ rbenv shell truffleruby-1.0.0-rc14
$ ruby -v
truffleruby 1.0.0-rc14, like ruby 2.6.2, GraalVM CE Native [x86_64-darwin]
```

A screenshot of a Mac OS X application window titled "demo.rb". The window contains a code editor with the following Ruby script:

```
1 require 'erb'
2
3 template = ERB.new(%{
4     <h1>Hello world!</h1>
5     <p>The time is <%= now %></p>
6 })
7
8 loop do
9     start = Time.now
10
11    100_000.times do
12        now = Time.now
13        puts template.result(binding)
14    end
15
16    $stderr.puts Time.now - start
17 end
18
```

The code uses ERB templating to generate an HTML document with a header and a timestamp. It then performs 100,000 iterations to demonstrate performance.

The status bar at the bottom shows "Line 1, Column 1", "Spaces: 2", and "Ruby".

# TruffleRuby compared to MRI

# Like MRI, TruffleRuby

- runs standard Ruby code
- uses RubyGems
- can be distributed as a binary tarball
- runs in Ruby version managers like rbenv, RVM, chruby
- doesn't need a JVM to run
- starts instantly
- supports C extensions
- is open source

# Compared to MRI, TruffleRuby has

- a concurrent, compacting GC
- an optimizing just-in-time native code compiler (a JIT)
- runs that run in parallel
- optimised data structures
- a graphical debugger
- more of the implementation in Ruby rather than C
- support for interop with Java, JavaScript, Python, R, and other languages

# TruffleRuby compared to JRuby

# Like JRuby, TruffleRuby

- can run on the JVM
- uses the GC and JIT from the JVM, but in a different way
- runs threads in parallel
- is partially implemented in Java
- can use Java libraries
- is open source

# Compared to JRuby, TruffleRuby

- can run without the JVM
- can start instantly
- can be embedded in a native application
- has optimized data structures
- can run C extensions
- has a more powerful JIT
- has more of the implementation in Ruby rather than Java
- supports a standard Ruby stack

# The big idea behind TruffleRuby

1000

100

10

1

mean



C

C++

JAVA

JavaScript

perl

php

python

Ruby



## Current situation

## How it should be

Prototype a new language

Parser and language work to build syntax tree (AST), AST Interpreter

Write a “real” VM

In C/C++, still using AST interpreter, spend a lot of time implementing runtime system, GC, ...

People start using it

People complain about performance

Define a bytecode format and write bytecode interpreter

Performance is still bad

Write a JIT compiler  
Improve the garbage collector

Prototype a new language in Java

Parser and language work to build syntax tree (AST)  
Execute using AST interpreter

People start using it

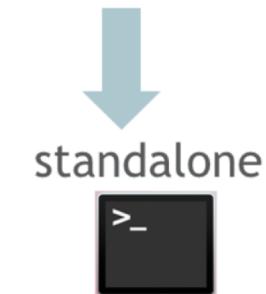
And it is already fast



Automatic transformation of interpreters to compiler

# GraalVM™

Embeddable in native or managed applications



# How TruffleRuby works

# TruffleRuby is implemented with

- a core in Java providing basic functionality
- a layer on top of that implemented in Ruby for most functionality
- most of the standard library from MRI
- RubyGems, Bundler, etc unmodified
- All similar to the original idea of Rubinius

The screenshot shows the Truffleruby IDE interface. On the left, a sidebar lists various Ruby files with their paths. The file 'integer.rb' is currently selected and highlighted in grey. The main workspace on the right displays the content of the 'integer.rb' file, which contains methods for floating-point division and character encoding conversion. The code is color-coded for readability, and line numbers are visible on the left side of the editor.

```
/* file.rb
/* file_test.rb
/* float.rb
/* gc.rb
/* hash.rb
/* immediate.rb
/* integer.rb
/* io.rb
/* kernel.rb
/* main.rb
/* marshal.rb
/* math.rb
/* method.rb
/* module.rb
/* mutex.rb
/* nil.rb
/* numeric.rb
/* object_space.rb
/* posix.rb
/* post.rb
/* pre.rb
/* proc.rb
/* process.rb
/* random.rb
/* range.rb
/* rational.rb
/* rationalizer.rb
/* regexp.rb

integer.rb

82
83     def fdiv(n)
84         if n.kind_of?(Integer)
85             to_f / n
86         else
87             redo_coerced :fdiv, n
88         end
89     end
90
91     def times
92         return to_enum(:times) { self } unless block_given?
93
94         i = 0
95         while i < self
96             yield i
97             i += 1
98         end
99         self
100    end
101
102    def chr(enc=undefined)
103        if self < 0 || (self & 0xffff_ffff) != self
104            raise RangeError, "#{self} is outside of the valid char
105        end
106
107        if undefined.equal? enc
108            if 0xff < self
109                enc = Encoding.default_internal
110                if enc.nil?
111                    raise RangeError, "#{self} is outside of the valid c
112                end
113            elsif self < 0x80
114                enc = Encoding::US_ASCII
115            else
116                Encoding::ASCII_8BIT
```

The screenshot shows an IDE interface with the following details:

- Title Bar:** truffleruby [~/Documents/truffleruby/truffleruby] - .../src/main/java/org/truffleruby/core/numeric/IntegerNodes.java [org.truffleruby]
- Toolbar:** Includes icons for file operations, search, and Git integration.
- Project Explorer (Left):** Shows the project structure under "1: Project". The "org.truffleruby.java" folder is expanded, showing subfolders like "algorithms", "aot", "builtins", "cext", "collections", "core", "debug", "extra", and "ffi". Within "extra", "PointerNodes" is selected and highlighted in blue.
- Code Editor (Center):** Displays the content of `IntegerNodes.java`. The code defines an abstract class `AddNode` that extends `BignumCoreMethodNode`. It contains several specialization methods for addition, including methods for int, long, double, and DynamicObject inputs. The code uses annotations like `@CoreMethod`, `@Specialization`, and `@Cached`.
- Toolbars and Status Bar (Bottom):** Includes tabs for "TODO", "Version Control", and "Terminal". The status bar shows "Dockerfile detection: You may setup Docker deployment run configuration for the following file(s): tool/docker/Dockerfile // Disable this notification (yesterday 19:04)" and other system information like date and time.



A screenshot of a dark-themed code editor window titled "add.rb". The file contains the following code:

```
1 puts 14 + 2
```

The editor has a status bar at the bottom with the following information:

- Line 1, Column 6; Saved ~/add.rb (UTF-8)
- Spaces: 2
- Gilles Duboscq<sup>†</sup> Christian Humer<sup>†</sup> Gregor Richards<sup>§</sup> Doug Simon<sup>\*</sup> Mario Wolczko<sup>\*</sup>

On the right side of the status bar, there are links for "File Them All" and "File Them All".

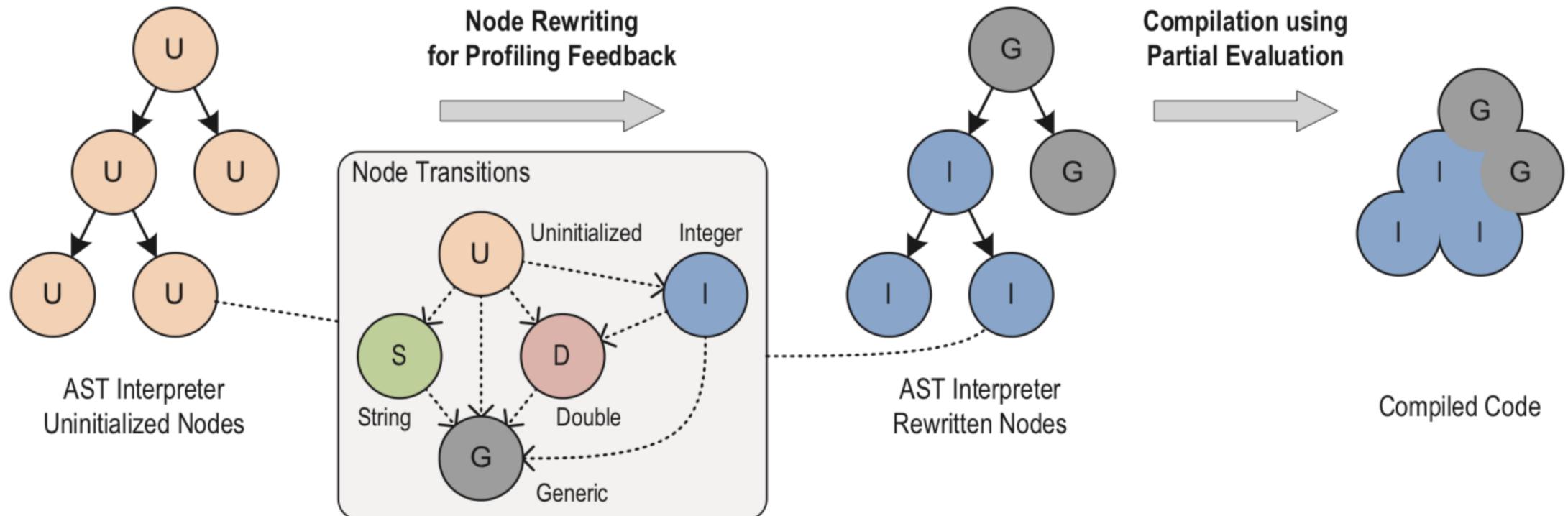


chrisseaton — bash — 132x33

```
[Chriss-MacBook-Pro:~ chrisseaton$ ruby --dump=parsetree add.rb
#####
## Do NOT use this node dump for any purpose other than ##
## debug and research. Compatibility is not guaranteed. ##
#####

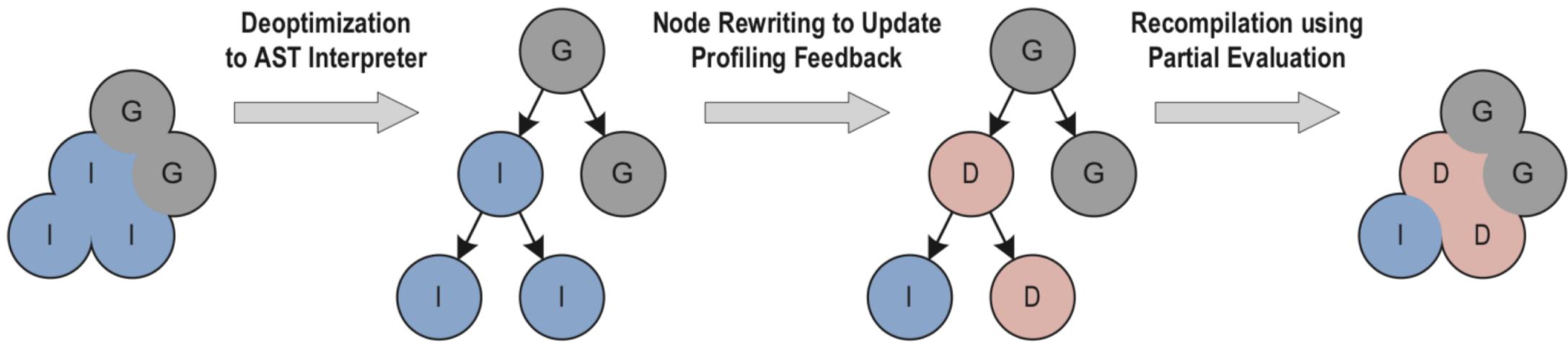
# @ NODE_SCOPE (line: 1, location: (1,0)-(1,11))
# +- nd_tbl: (empty)
# +- nd_args:
# | (null node)
# +- nd_body:
#   @ NODE_FCALL (line: 1, location: (1,0)-(1,11))*
#   +- nd_mid: :puts
#   +- nd_args:
#     @ NODE_ARRAY (line: 1, location: (1,5)-(1,11))
#     +- nd_alen: 1
#     +- nd_head:
#       @ NODE_OPCALL (line: 1, location: (1,5)-(1,11))
#       +- nd_mid: :=
#       +- nd_recv:
#         @ NODE_LIT (line: 1, location: (1,5)-(1,7))
#         +- nd_lit: 14
#         +- nd_args:
#           @ NODE_ARRAY (line: 1, location: (1,10)-(1,11))
#           +- nd_alen: 1
#           +- nd_head:
#             @ NODE_LIT (line: 1, location: (1,10)-(1,11))
#             +- nd_lit: 2
#             +- nd_next:
#               (null node)
#             +- nd_next:
#               (null node)
Chriss-MacBook-Pro:~ chrisseaton$ ]
```

Giles Duboscq Christian Turner Gregor Richards Doug Simon Mario Wolczko\*



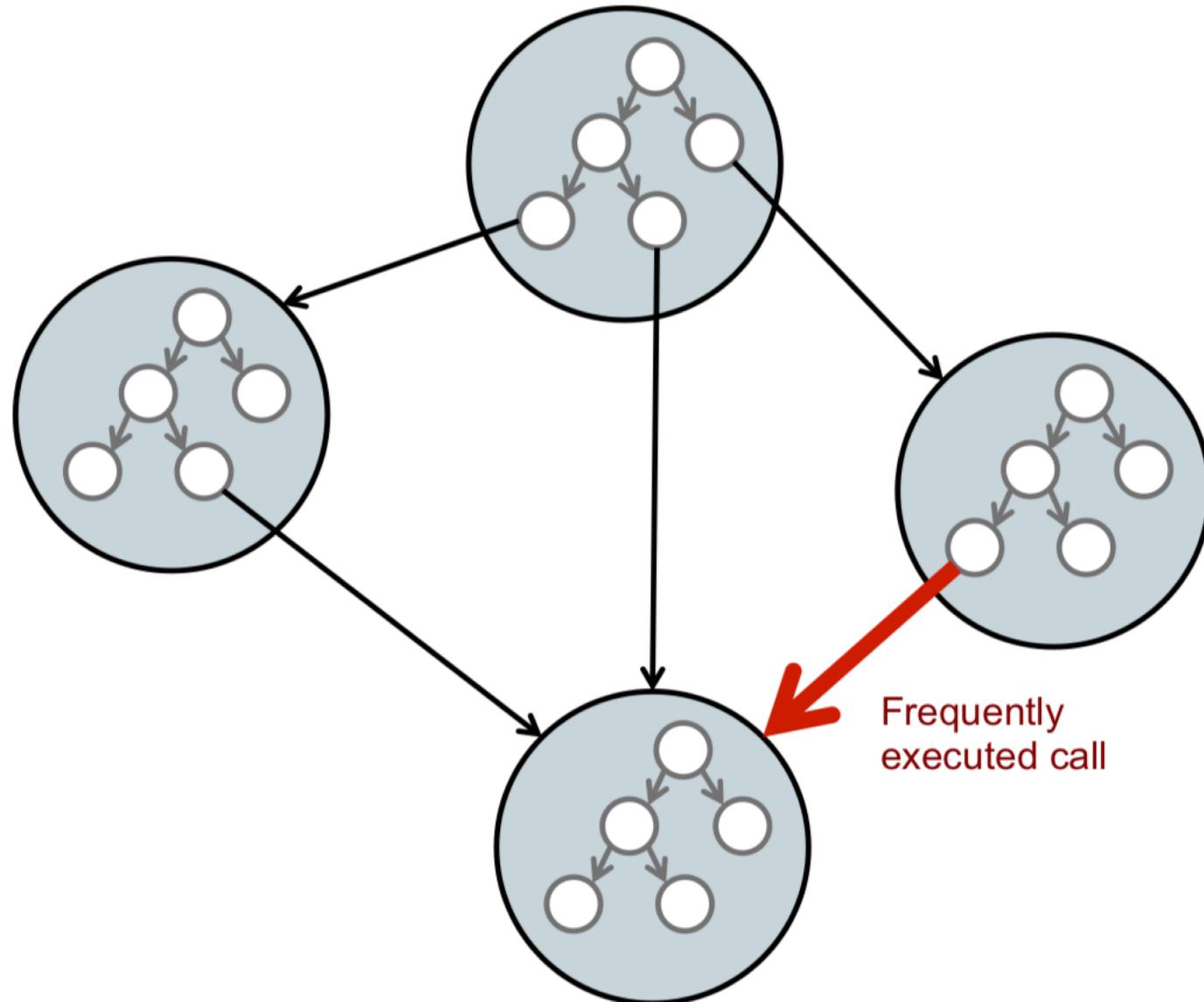
### One VM to Rule Them All

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

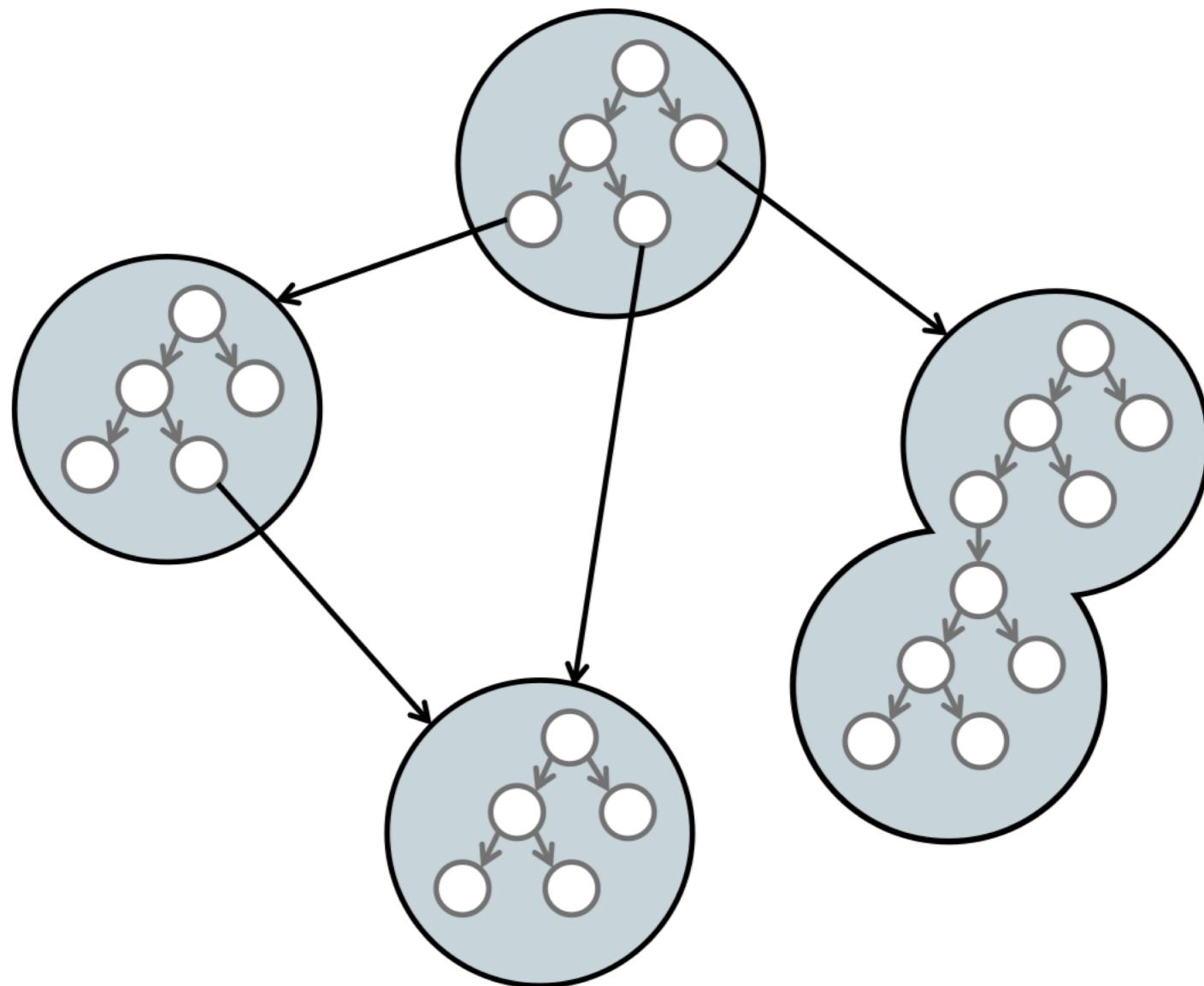


**One VM to Rule Them All**

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

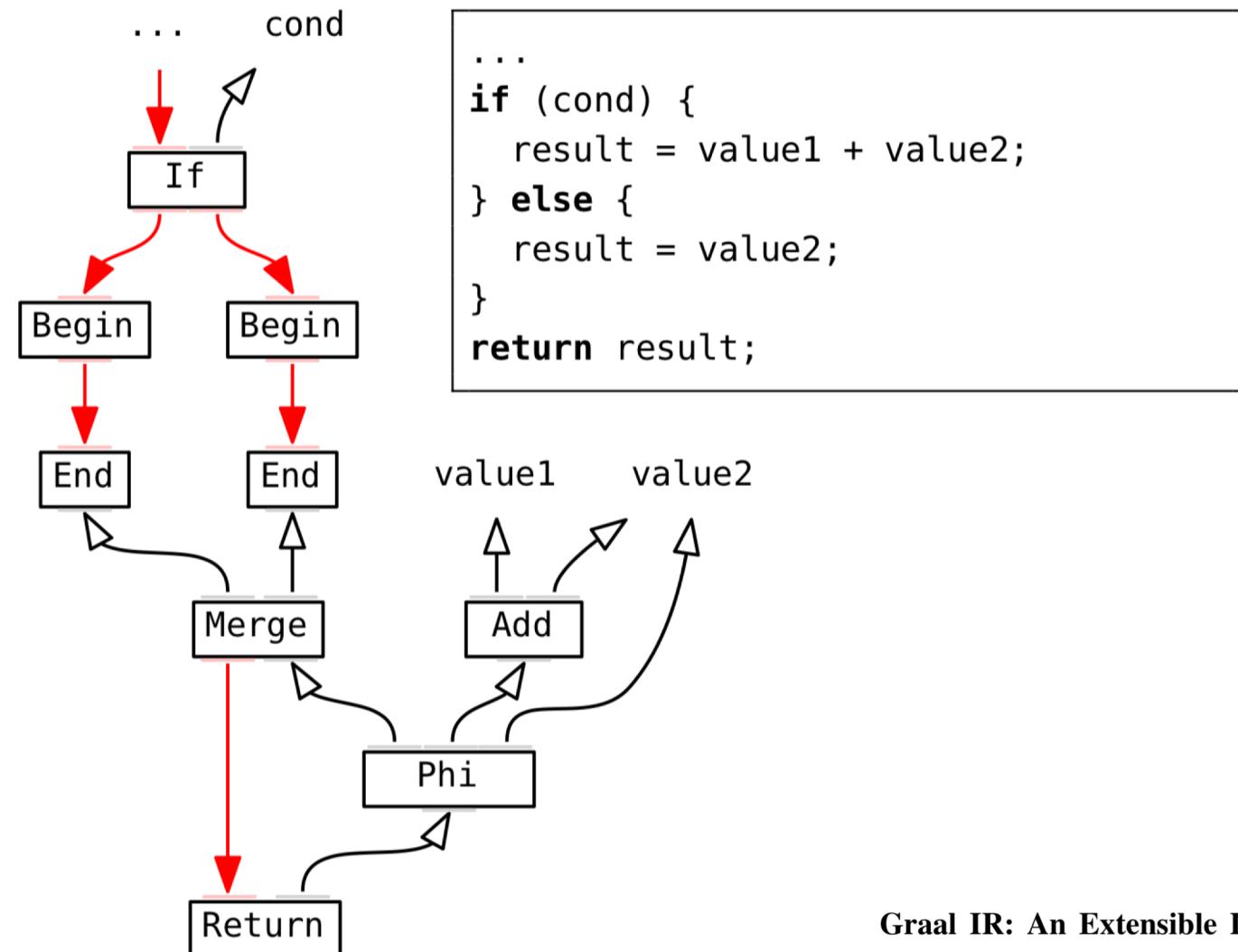


Frequently  
executed call



# What is Graal?

- A compiler from Java bytecode\* to native machine code
  - Not a compiler from Java source code to Java bytecode
  - ‘Java bytecode’ could come from Truffle
- It’s a compiler library rather than being an executable, like GCC
- Some things I didn’t tie Graal down to there:
  - I didn’t say it was specifically a just-in-time compiler
  - I didn’t say the bytecode had to come directly from a program as written



## Graal IR: An Extensible Declarative Intermediate Representation

Gilles Duboscq\* Lukas Stadler\* Thomas Würthinger†  
 Doug Simon† Christian Wimmer† Hanspeter Mössenböck\*

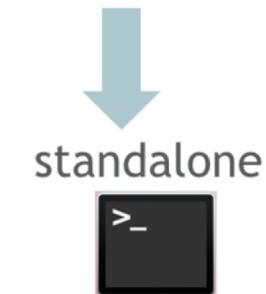
# Polyglot



Automatic transformation of interpreters to compiler

# GraalVM™

Embeddable in native or managed applications



A screenshot of a terminal window titled "poly.rb". The window contains the following Ruby code:

```
array = [1, 2, 3, 4, 5, 6]
p array.map { |n| n * 2 }
doubler = Polyglot.eval('python', %{lambda a: [n * 2 for n in a]})
p doubler.call(array)
```

The code demonstrates the use of the `Polyglot` gem to evaluate Python code within a Ruby script. The Python code defines a lambda function that doubles each element in an array. This lambda is then passed to the `Polyglot.eval` method, which executes it in a Python environment. The result is a Ruby object that is printed using the `p` command.



Line 1, Column 1

Spaces: 2

Ruby

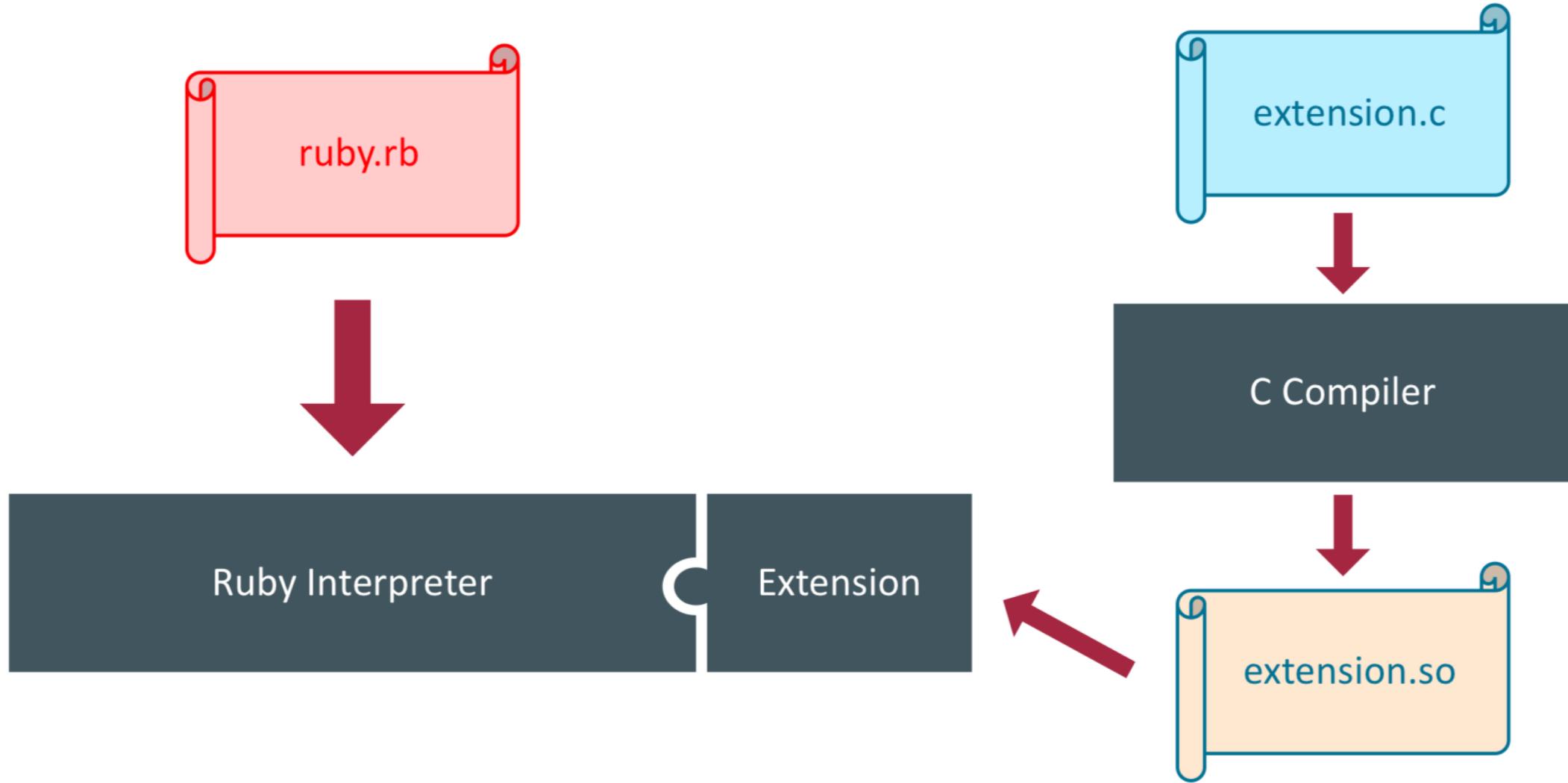
The screenshot shows the official GraalVM website at [graalvm.org](https://graalvm.org). The page has a dark teal header with navigation links for Home, Docs (which is highlighted), Downloads, Community, and social media icons for Twitter and GitHub. A star icon indicates 8,222 stars on GitHub. The main title "GraalVM™" is in large white font, followed by the tagline "Run Programs Faster Anywhere". Below the tagline are two buttons: "WHY GRAALVM" in orange and "GET STARTED" in white. To the right is a diagram illustrating GraalVM's polyglot nature, showing various application components (Java, MySQL, Standalone, OpenJDK, Node, Oracle Database) connected to a central "GraalVM" block.

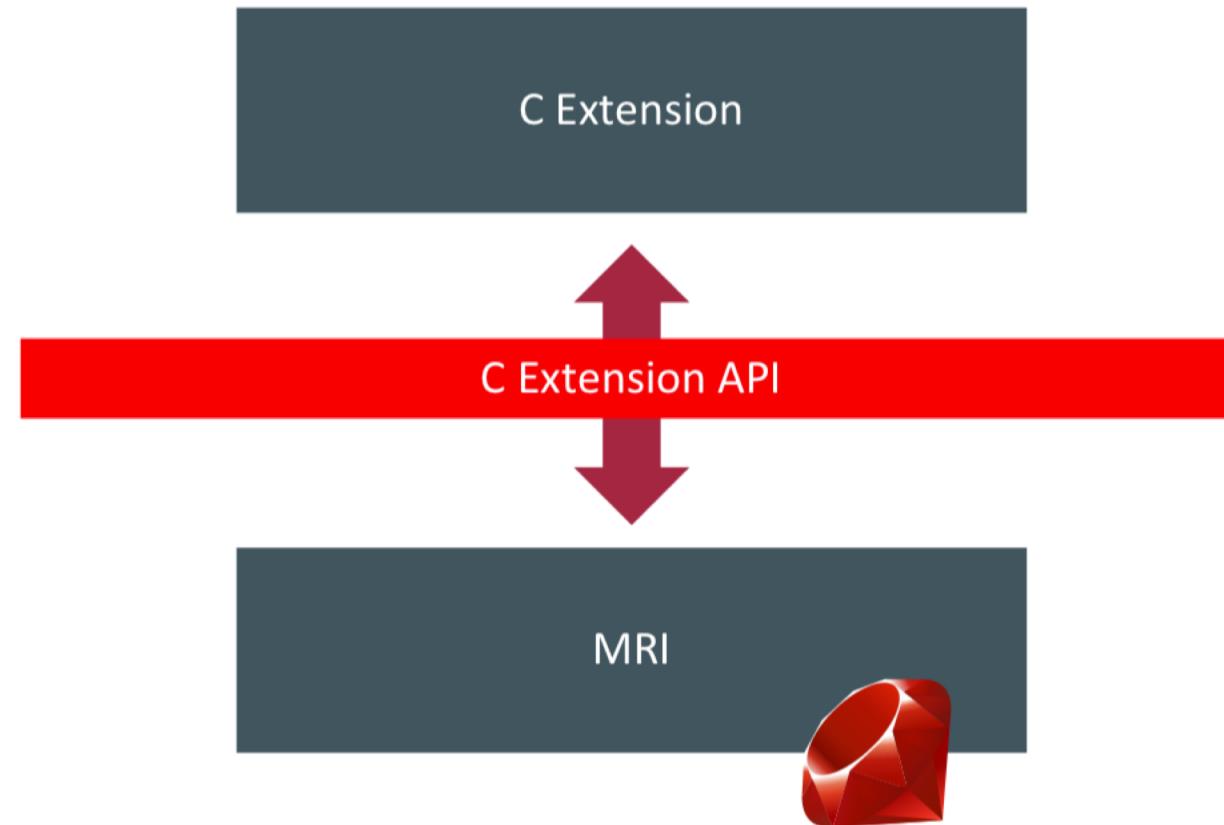
High-performance  
polyglot VM

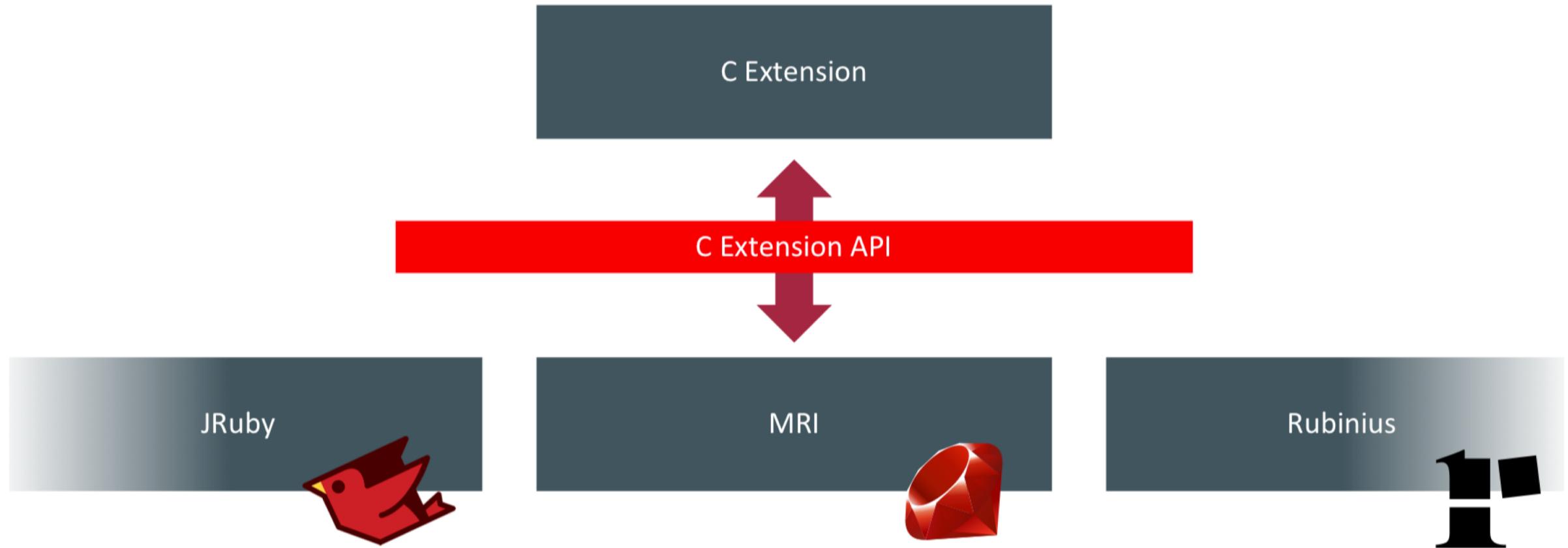
GraalVM is a universal virtual machine for running applications written in JavaScript, Python, Ruby, R, JVM-based languages like Java, Scala, Kotlin, Clojure, and LLVM-based languages such as C and C++.

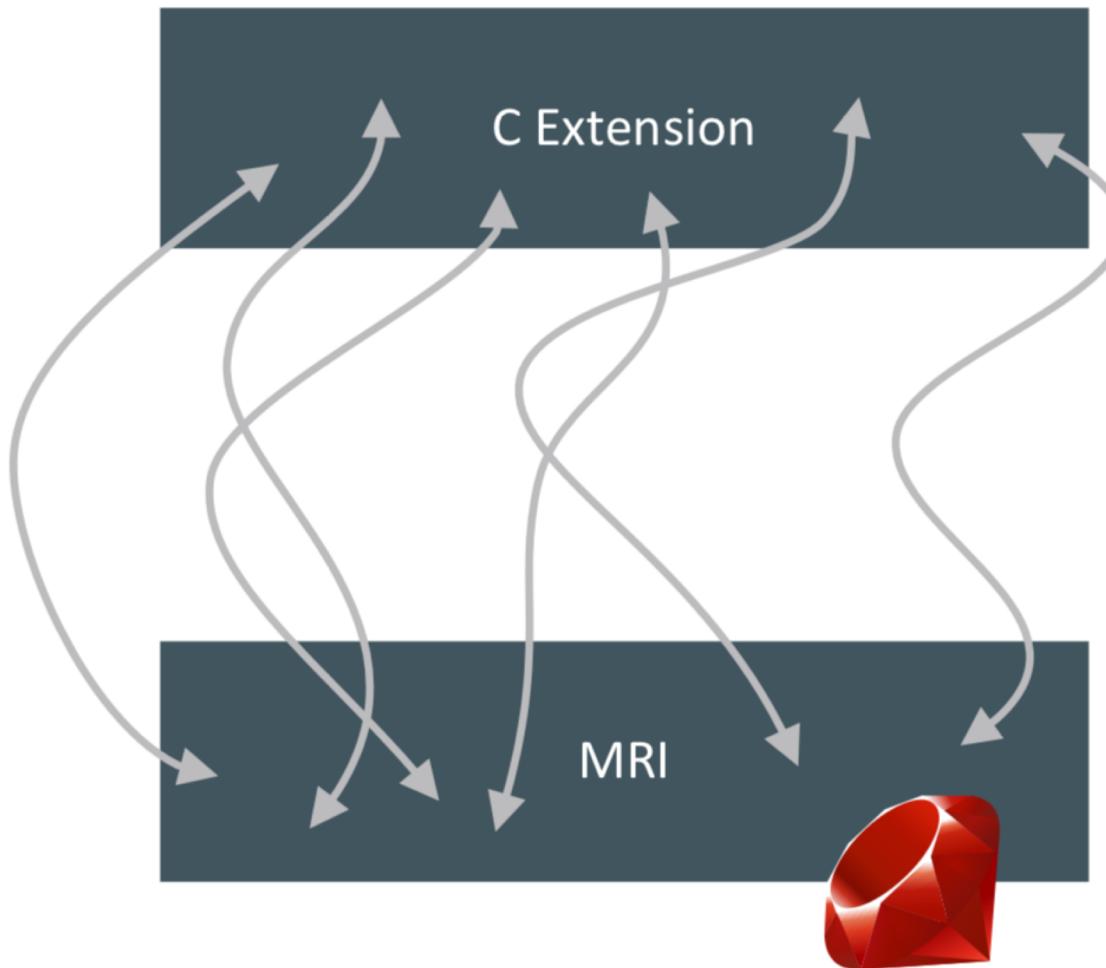
GraalVM removes the isolation between programming languages and enables interoperability in a shared

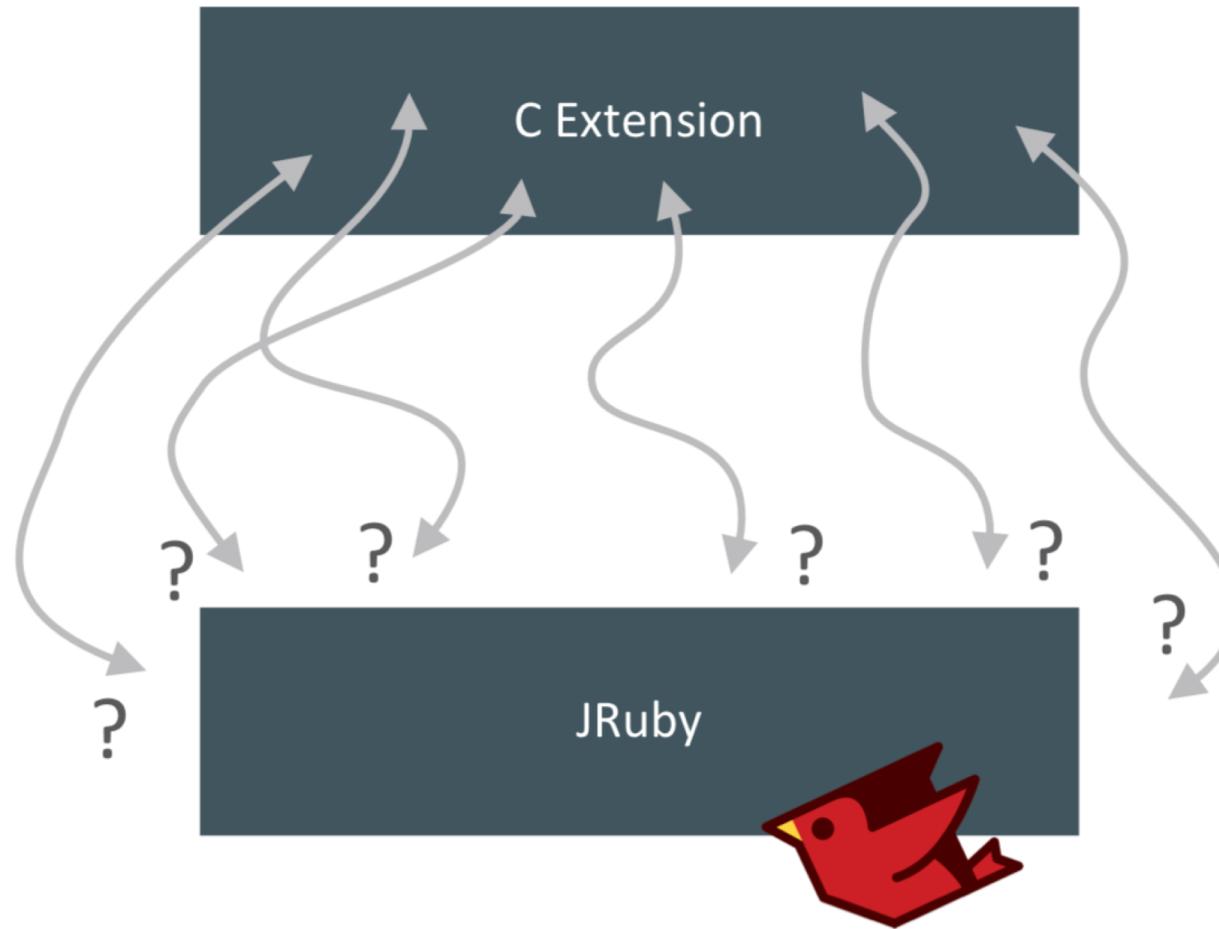
# Running C extensions

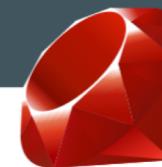
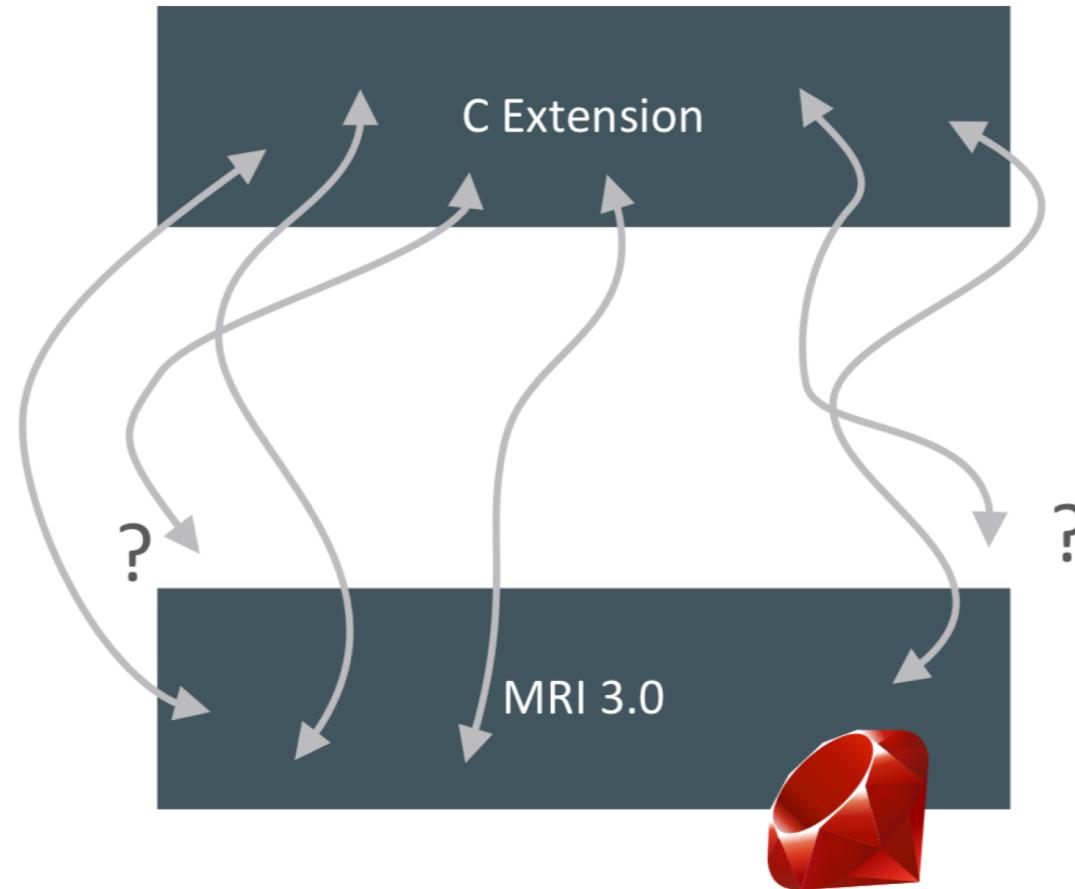


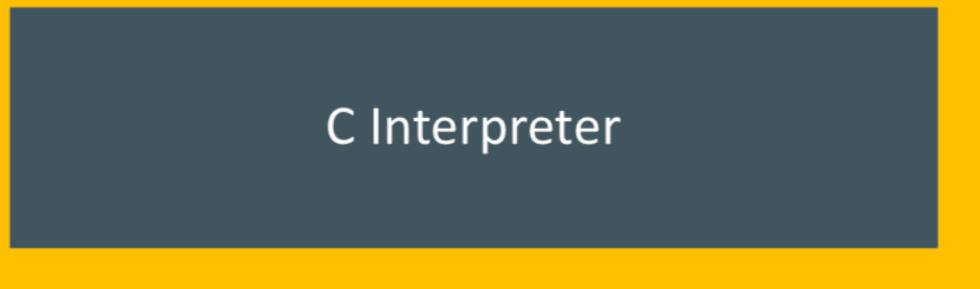
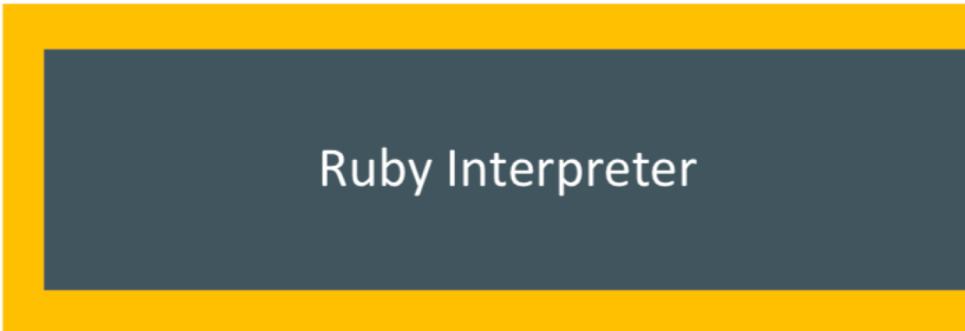


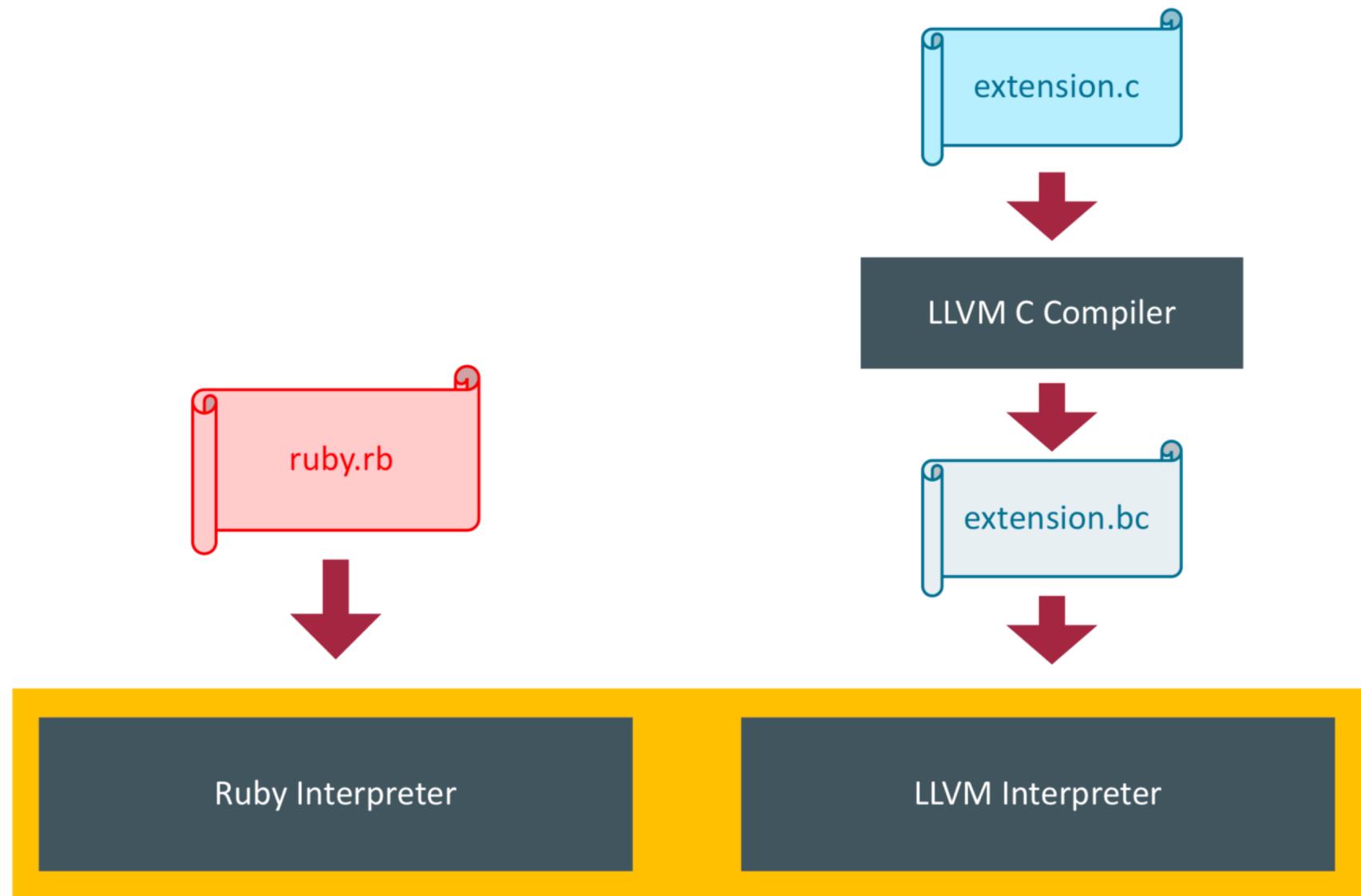


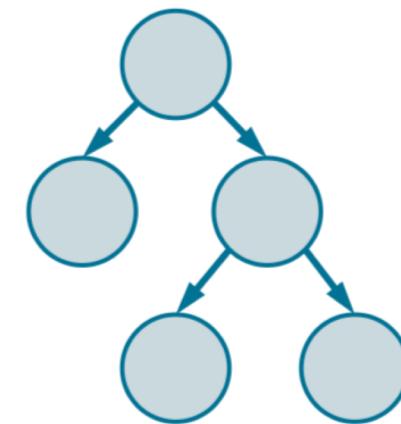
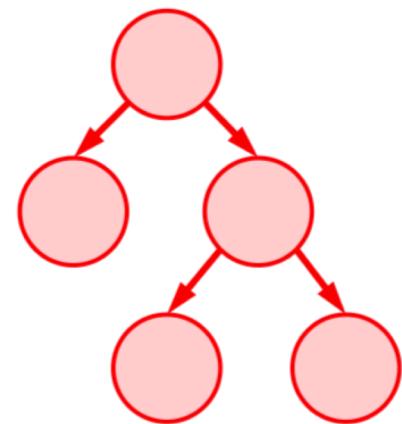




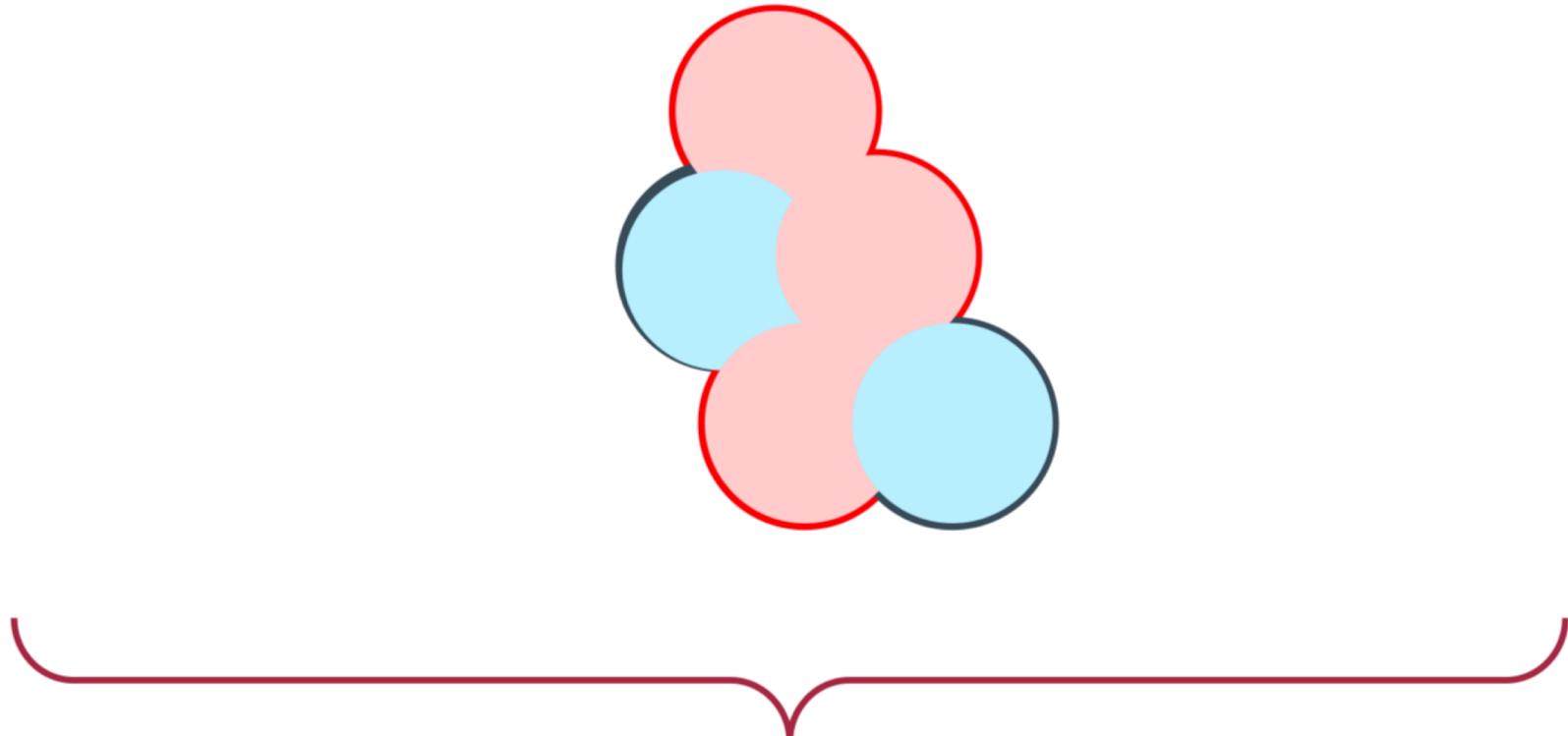








## Optimisations



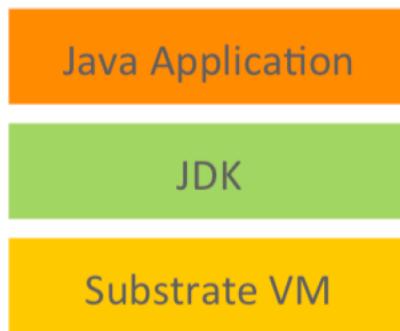
Optimisations

# Native compilation

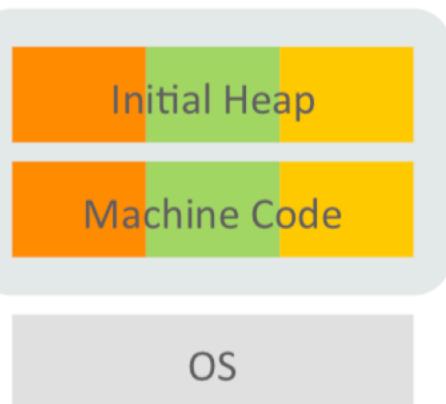
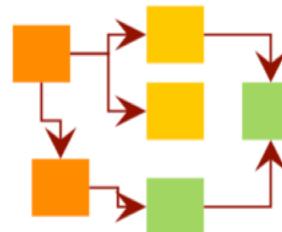
# The JVM can be a bit of a burden

- required to run JRuby
- installed separately
- updated separately
- starts relatively slowly
- uses relatively large amount of memory

## Static Analysis



## Ahead-of-Time Compilation



All Java classes from application, JDK, and Substrate VM

Reachable methods, fields, and classes

Application running without compilation or class loading

# Sum up

# 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 Hummer  
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

# Questions

# Safe Harbor Statement

The preceding 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. Oracle reserves the right to alter its development plans and practices at any time, and 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.

# Integrated Cloud Applications & Platform Services

**ORACLE®**