ORACLE®

ORACLE

One VM to Rule Them All

Christian Wimmer, Chris Seaton

VM Research Group, Oracle Labs

MANCHESTER 1824

The University of Manchester

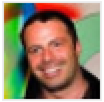
The following is intended to provide some insight into a line of research in Oracle Labs. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing 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.

 Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

One Language to Rule Them All?

Let's ask Google…

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



venturebeat.com/2011/.../javascript-**one-language-to-rule-them**-...

by Peter Yared - in 23 Google+ circles

Jul 29, 2011 - Why code in two different scripting languages, one on the client and one on the server? It's time for **one language to rule them all**. Peter Yared **...**

[PDF] Python: **One** Script (**Language**) **to rule them all** - Ian Darwin

www.darwinsys.com/python/python4unix.pdf

Another **Language**? ▸ Python was invented in 1991 by Guido van. Rossum. ▪ Named after the comedy troupe, not the snake. ▸ Simple. ▪ They **all** say that!

Q & Stuff: **One Language to Rule Them All - Java**

qstuff.**blogspot**.com/2005/10/**one-language-to-rule-them-all-java**.html

Oct 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** ...

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

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

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

ORACLE

"Write Your Own Language"

Current situation	How it should be

Current situation

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

How it should be

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

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Truffle Requirements

Ruby, JavaScript, Python, R, J, Java, Groovy, Clojure, Scala ...

Simplicity

+

Generality

+

Performance

```
@Specialization(
   rewriteOn=ArithmeticException.class)
int add(int l, int r) {
  return Math.addExact(l, r);
}

@Specialization
double add(double l, double r) {
  return l + r;
}

@Specialization(guards = "isString")
String doString(Object l, Object r) {
  return l.toString() + r.toString();
}
```

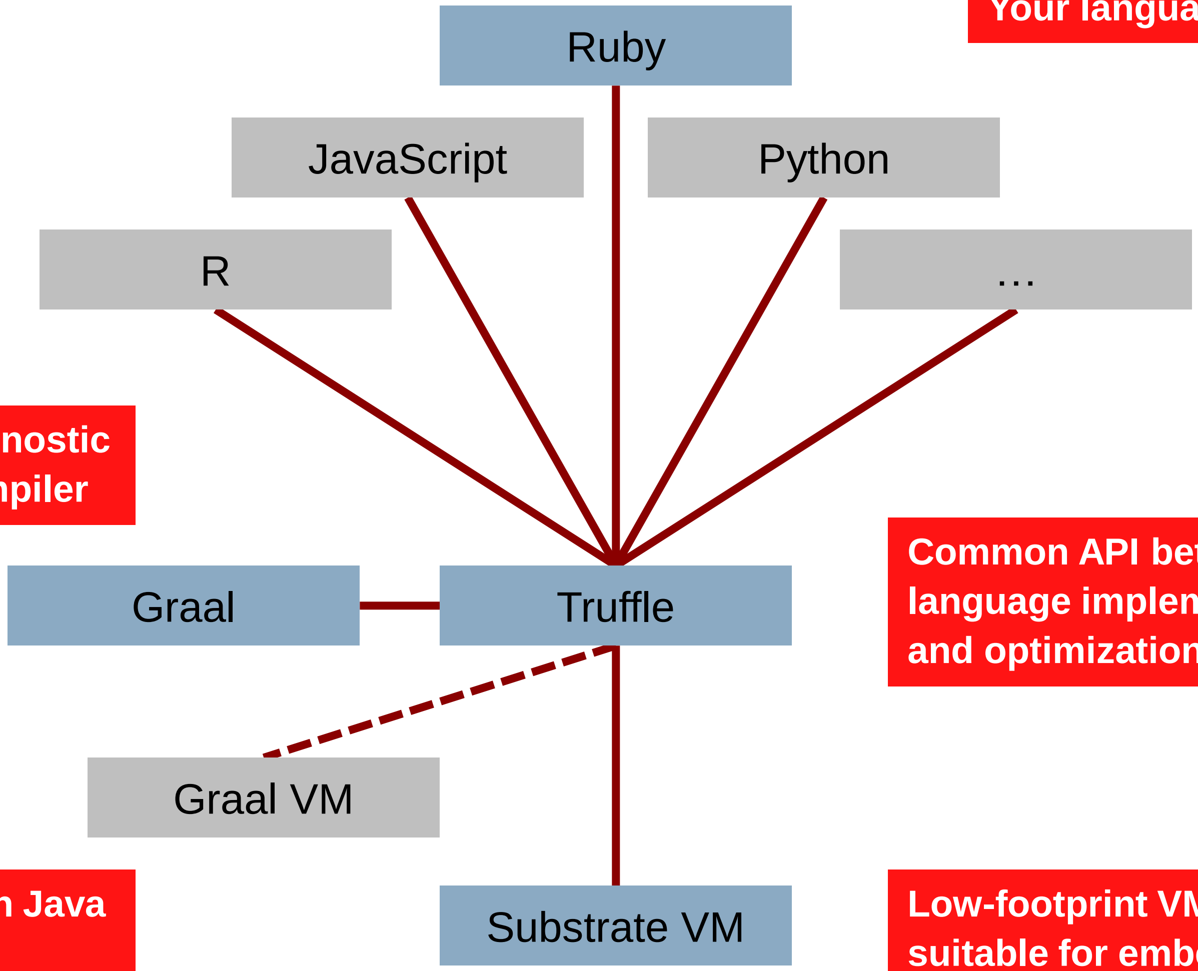
```
function f(a, n) {
  var x = 0;
  while (n-- > 0) {
    x = x + a[n];
  }
  return x;
}
```

```
L1: decl rax
jz L2
movl rcx, rdx[16+4*rax]
cvtsi2sd xmm1, rcx
addsd xmm0, xmm1
jmp L1
L2:
```


Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

System Structure

Your language here!



Language agnostic dynamic compiler

Common API between language implementation and optimization system

Integrate with Java applications

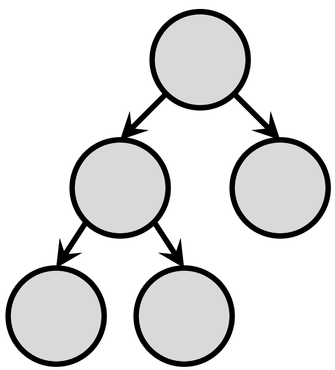
Low-footprint VM, also suitable for embedding

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

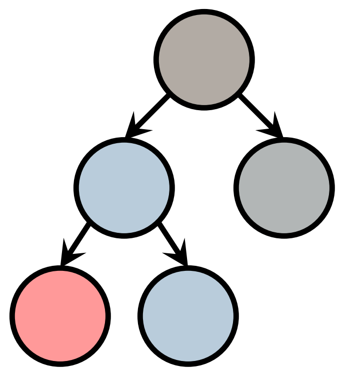
Truffle Approach

AST Rewriting for Type Feedback

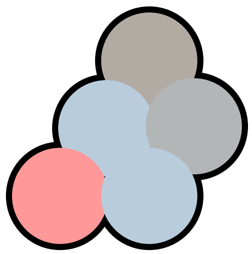
Automatic Partial Evaluation



AST Interpreter
Uninitialized Nodes



AST Interpreter
Rewritten Nodes



Compiled Code

- Eliminate boxing of primitive values
- Eliminate dynamic type checks
- AST Inlining

- Syntax tree nodes are “stable”
- Aggressive constant folding, method inlining, escape analysis
- Deoptimize compiled code on tree rewrite

 Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

More Details on Truffle

Accepted for Onward! 2013, October 26-31 2013, Indianapolis, IN

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 §S³ Lab, Purdue University

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

Abstract

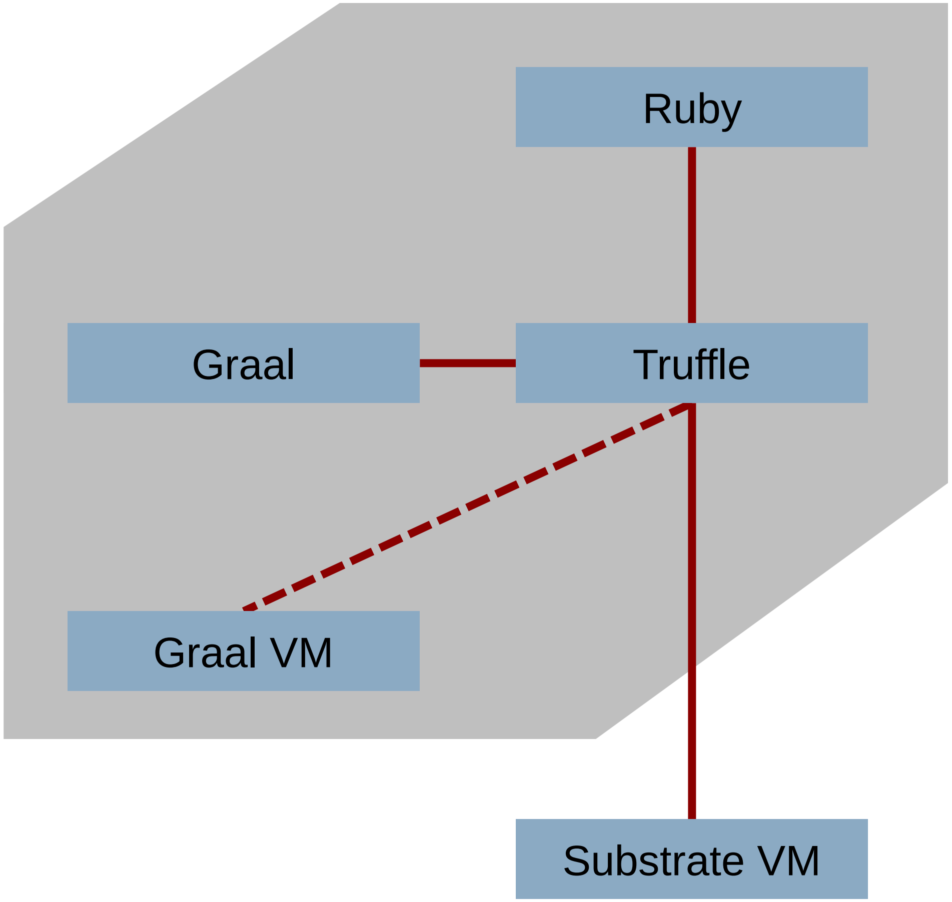
Building high-performance virtual machines is a complex and expensive undertaking; many popular languages still have low-performance implementations. We describe a new approach to virtual machine (VM) construction that amortizes much of the effort in initial construction by allowing new languages to be implemented with modest additional effort. The approach relies on abstract syntax tree (AST) interpretation where a node can rewrite itself to a more specialized or more general node, together with an optimizing compiler that exploits the structure of the interpreter. The compiler uses speculative assumptions and deoptimization in order to produce efficient machine code. Our initial experience suggests that high performance is attainable while preserving a modular and layered architecture, and that new high-performance language implementations can be obtained by writing little more than a stylized interpreter.

as Microsoft's Common Language Runtime, the VM of the .NET framework [43]. These implementations can be characterized in the following way:

- Their performance on typical applications is within a small integer multiple (1-3x) of the best statically compiled code for most equivalent programs written in an unsafe language such as C.
- They are usually written in an unsafe, systems programming language (C or C++).
- Their implementation is highly complex.
- They implement a single language, or provide a bytecode interface that preferentially advantages a narrow set of languages to the detriment of other languages.

In contrast, there are numerous languages that are popular, have been around for about 20 years, and yet still have

Ruby Prototype: High Performance

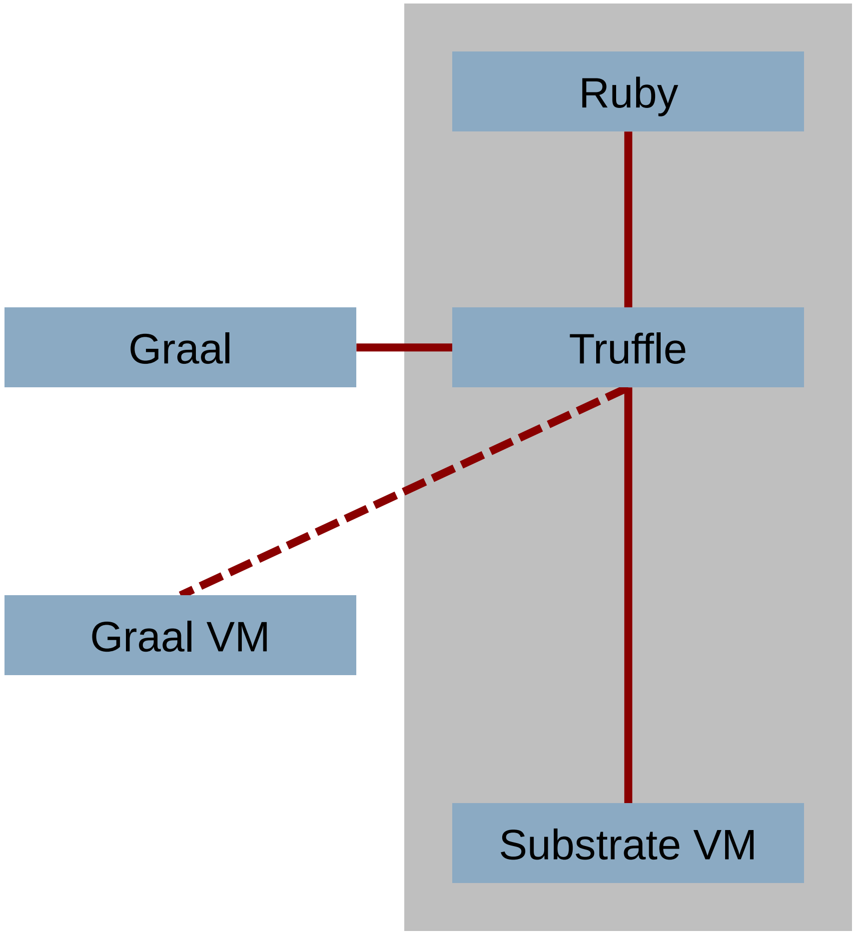


Fastest Ruby implementation …

… for the few benchmarks that we looked at

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Ruby Prototype: Low Footprint



Startup time (“Hello World”) comparable to MRI

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Ruby Prototype: Completeness

- RubySpec
  - A library of executable assertions that covers the language, core library and standard library
  - This is the defacto Ruby spec
  - Gives us a quantifiable result for how much of Ruby we implement correctly

Over 45% of RubySpec

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Completeness

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Completeness: Informally

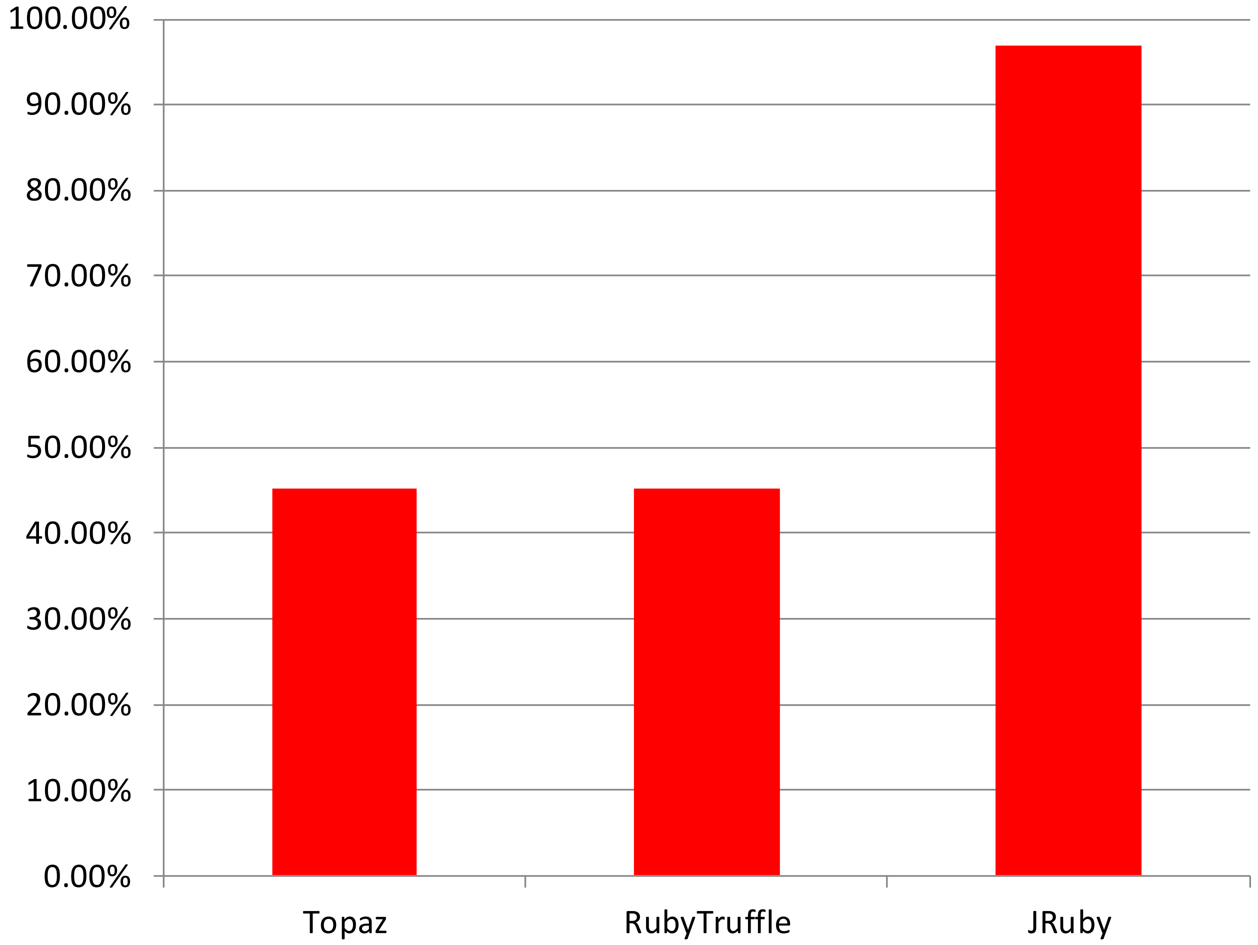
Language Feature	Implemented	Notes
Fixnum to Bignum promotion	✓	
Support for floating point	✓	
Closures	✓	
Bindings and eval	✓	
callcc and Continuation	✓	Very limited support, the same as JRuby
Fibers	✓	Slightly limited support, the same as JRuby
Frame local variables	✓	
C extensions		
Ruby 1.9 encoding	✓	
Garbage collection	✓	
Concurrency and parallelism	✓	We currently use a GIL
Tracing and debugging	✓	
ObjectSpace	✓	
Method invalidation	✓	
Constant invalidation	✓	
Ruby on Rails		

Charles Nutter: 'So You Want to Optimize Ruby' http://blog.headius.com/2012/10/so-you-want-to-optimize-ruby.html

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Completeness: More formally via RubySpec

Running language tests



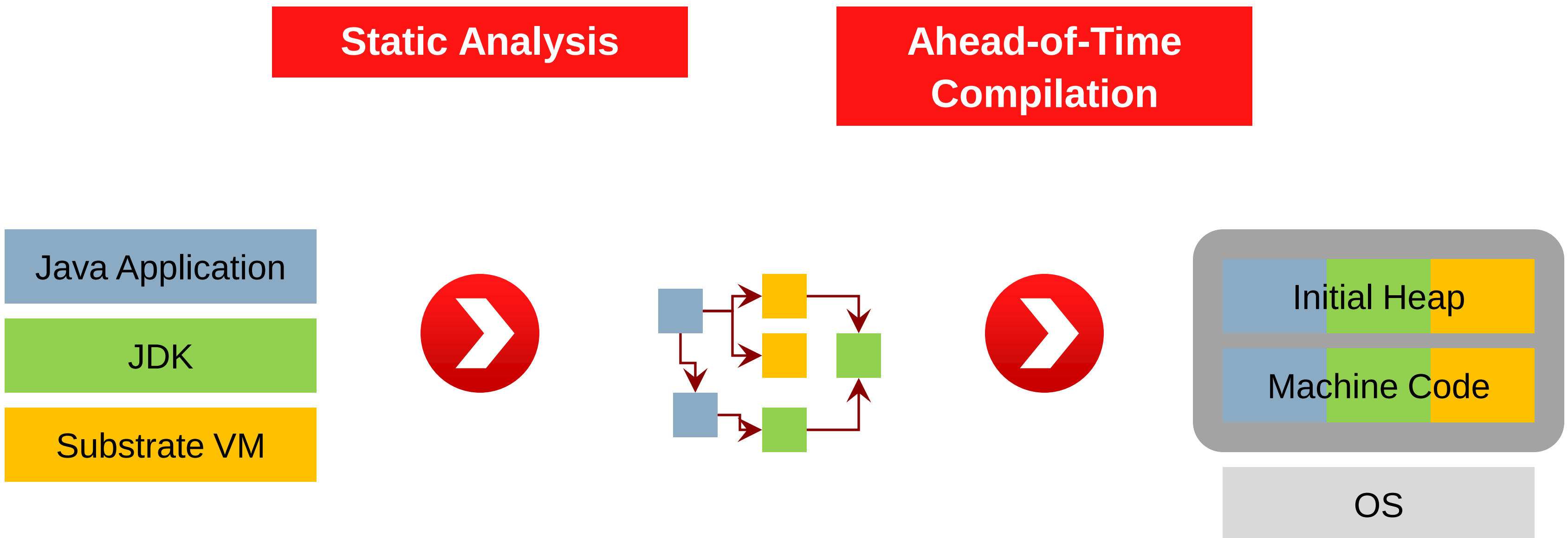
ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Low Footprint

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Substrate VM Execution Model



All Java classes from application, JDK, and Substrate VM

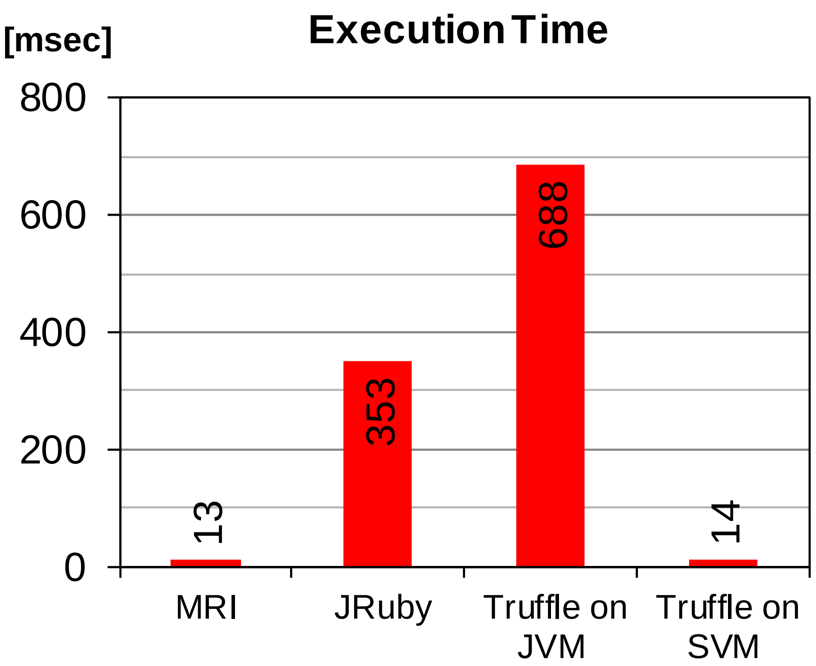
Reachable methods, fields, and classes

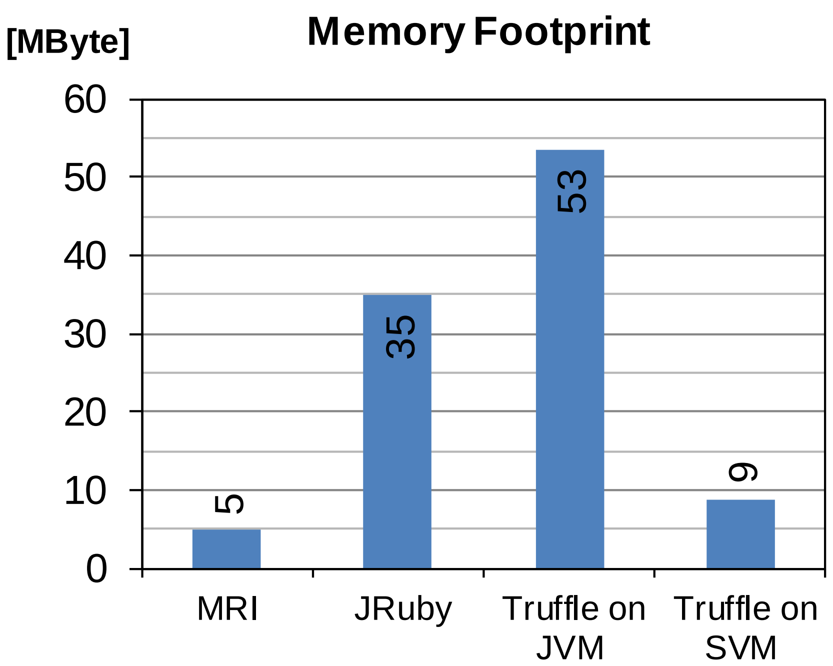
Application running without compilation or class loading

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Startup Performance

Running "Hello World"





Execution time: `time -f "%e"`

Memory footprint: `time -f "%M"`

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

High Performance

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Why is Ruby Slow?

```
-b + (Math.sqrt(b**2 - 4*a*c)) / 2*a
```


Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Why is Ruby Slow?

-b + (Math.sqrt(b**2 - 4*a*c)) / 2*a

execute b
check that b is a Float
check that the negate method in Float has not changed
calculate negation
check the result of that is a Float
execute b
check that b is a Float
check that the power method in Float has not changed
calculate power
check the result of that is a Float
execute a
check that a is a Float
check that the multiply method in Float has not changed
calculate multiplication
check the result of that is a Float
execute c
check that c is a Float
check that the multiply method in Float has not changed
calculate multiplication
check the result of that is a Float
check that Math has not changed
check that the sqrt method in Math has not changed
calculate sqrt
check the result of that is a Float
execute a
check that a is a Float
check that the multiply method in Float has not changed
calculate multiplication
check the result of that is a Float
check that the division method in Float has not changed
calculate division

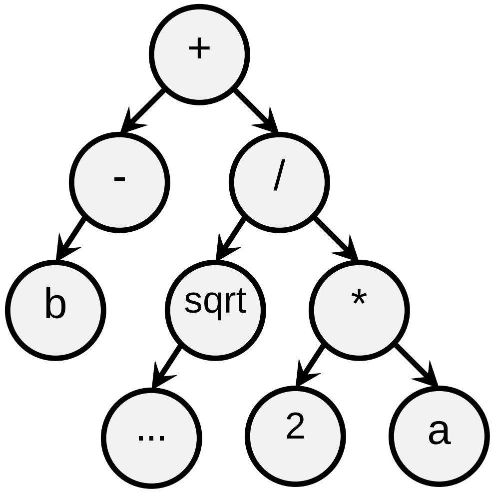
ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Improving Performance Using Truffle

-b + (Math.sqrt(b**2 - 4*a*c)) / 2*a

execute b
check that b is a Float
check that the negate method in Float has not changed
calculate negation
check the result of that is a Float
execute b
check that b is a Float
check that the power method in Float has not changed
calculate power
check the result of that is a Float
execute a
check that a is a Float
check that the multiply method in Float has not changed
calculate multiplication
check the result of that is a Float
execute c
check that c is a Float
check that the multiply method in Float has not changed
calculate multiplication
check the result of that is a Float
check that Math has not changed
check that the sqrt method in Math has not changed
calculate sqrt
check the result of that is a Float
execute a
check that a is a Float
check that the multiply method in Float has not changed
calculate multiplication
check the result of that is a Float
check that the division method in Float has not changed
calculate division



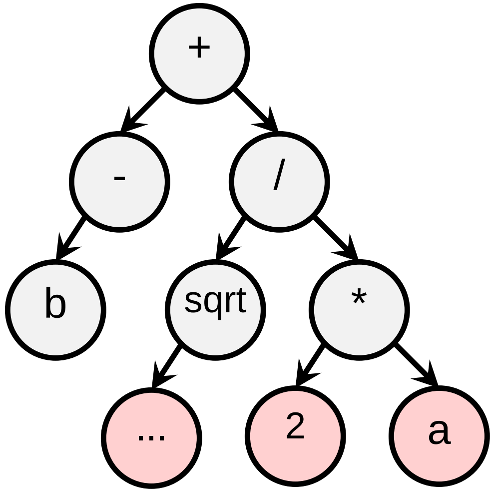
ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Improving Performance Using Truffle

-b + (Math.sqrt(b**2 - 4*a*c)) / 2*a

execute b
check that b is a Float
check that the negate method in Float has not changed
calculate negation
check the result of that is a Float
execute b
check that b is a Float
check that the power method in Float has not changed
calculate power
check the result of that is a Float
execute a
check that a is a Float
check that the multiply method in Float has not changed
calculate multiplication
check the result of that is a Float
execute c
check that c is a Float
check that the multiply method in Float has not changed
calculate multiplication
check the result of that is a Float
check that Math has not changed
check that the sqrt method in Math has not changed
calculate sqrt
check the result of that is a Float
execute a
check that a is a Float
check that the multiply method in Float has not changed
calculate multiplication
check the result of that is a Float
check that the division method in Float has not changed
calculate division

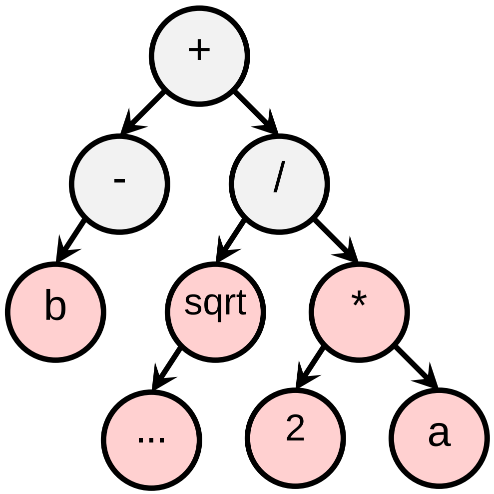


Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Improving Performance Using Truffle

-b + (Math.sqrt(b**2 - 4*a*c)) / 2*a

execute b
check that b is a Float
check that the negate method in Float has not changed
calculate negation
check the result of that is a Float
execute b
check that b is a Float
check that the power method in Float has not changed
calculate power
check the result of that is a Float
execute a
check that a is a Float
check that the multiply method in Float has not changed
calculate multiplication
check the result of that is a Float
execute c
check that c is a Float
check that the multiply method in Float has not changed
calculate multiplication
check the result of that is a Float
check that Math has not changed
check that the sqrt method in Math has not changed
calculate sqrt
check the result of that is a Float
execute a
check that a is a Float
check that the multiply method in Float has not changed
calculate multiplication
check the result of that is a Float
check that the division method in Float has not changed
calculate division



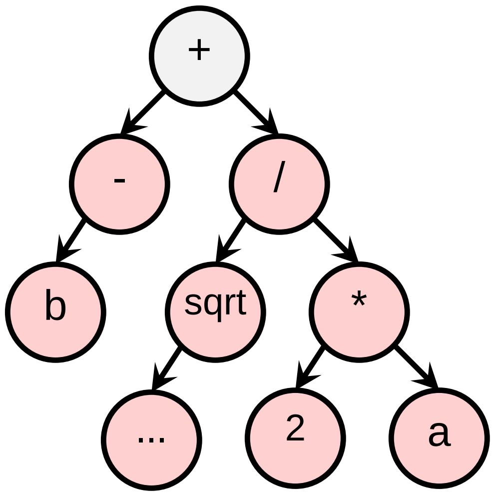
ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Improving Performance Using Truffle

-b + (Math.sqrt(b**2 - 4*a*c)) / 2*a

execute b
check that b is a Float
check that the negate method in Float has not changed
calculate negation
check the result of that is a Float
execute b
check that b is a Float
check that the power method in Float has not changed
calculate power
check the result of that is a Float
execute a
check that a is a Float
check that the multiply method in Float has not changed
calculate multiplication
check the result of that is a Float
execute c
check that c is a Float
check that the multiply method in Float has not changed
calculate multiplication
check the result of that is a Float
check that Math has not changed
check that the sqrt method in Math has not changed
calculate sqrt
check the result of that is a Float
execute a
check that a is a Float
check that the multiply method in Float has not changed
calculate multiplication
check the result of that is a Float
check that the division method in Float has not changed
calculate division

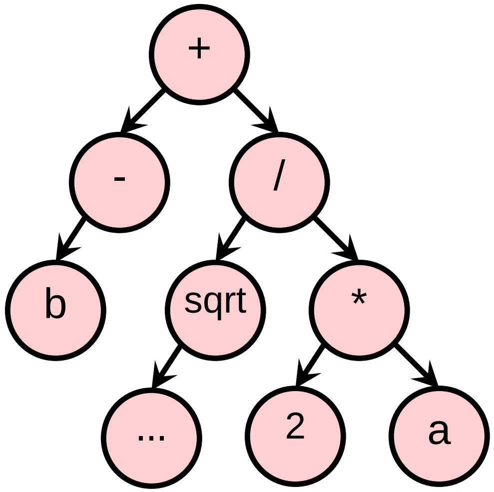


Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Improving Performance Using Truffle

-b + (Math.sqrt(b**2 - 4*a*c)) / 2*a

execute b
check that b is a Float
check that the negate method in Float has not changed
calculate negation
check the result of that is a Float
execute b
check that b is a Float
check that the power method in Float has not changed
calculate power
check the result of that is a Float
execute a
check that a is a Float
check that the multiply method in Float has not changed
calculate multiplication
check the result of that is a Float
execute c
check that c is a Float
check that the multiply method in Float has not changed
calculate multiplication
check the result of that is a Float
check that Math has not changed
check that the sqrt method in Math has not changed
calculate sqrt
check the result of that is a Float
execute a
check that a is a Float
check that the multiply method in Float has not changed
calculate multiplication
check the result of that is a Float
check that the division method in Float has not changed
calculate division

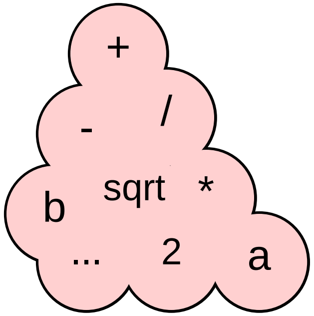


Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Improving Performance Using Truffle

-b + (Math.sqrt(b**2 - 4*a*c)) / 2*a

execute b
check that b is a Float
check that the negate method in Float has not changed
calculate negation
check the result of that is a Float
execute b
check that b is a Float
check that the power method in Float has not changed
calculate power
check the result of that is a Float
execute a
check that a is a Float
check that the multiply method in Float has not changed
calculate multiplication
check the result of that is a Float
execute c
check that c is a Float
check that the multiply method in Float has not changed
calculate multiplication
check the result of that is a Float
check that Math has not changed
check that the sqrt method in Math has not changed
calculate sqrt
check the result of that is a Float
execute a
check that a is a Float
check that the multiply method in Float has not changed
calculate multiplication
check the result of that is a Float
check that the division method in Float has not changed
calculate division



 Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Improving Performance Using Truffle

-b + (Math.sqrt(b**2 - 4*a*c)) / 2*a

execute b
check that b is a Float
check that the negate method in Float has not changed
calculate negation
check the result of that is a Float
execute b
check that b is a Float
check that the power method in Float has not changed
calculate power
check the result of that is a Float
execute a
check that a is a Float
check that the multiply method in Float has not changed
calculate multiplication
check the result of that is a Float
execute c
check that c is a Float
check that the multiply method in Float has not changed
calculate multiplication
check the result of that is a Float
check that Math has not changed
check that the sqrt method in Math has not changed
calculate sqrt
check the result of that is a Float
execute a
check that a is a Float
check that the multiply method in Float has not changed
calculate multiplication
check the result of that is a Float
check that the division method in Float has not changed
calculate division

execute b
check that the negate method in Float has not changed
calculate negation
execute b
check that the power method in Float has not changed
calculate power
execute a
check that the multiply method in Float has not changed
calculate multiplication
execute c
check that the multiply method in Float has not changed
calculate multiplication
check that Math has not changed
check that the sqrt method in Math has not changed
calculate sqrt
execute a
check that the multiply method in Float has not changed
calculate multiplication
check that the division method in Float has not changed
calculate division

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Improving Performance Using Graal

-b + (Math.sqrt(b**2 - 4*a*c)) / 2*a

execute b
check that the negate method in Float has not changed
calculate negation
execute b
check that the power method in Float has not changed
calculate power
execute a
check that the multiply method in Float has not changed
calculate multiplication
execute c
check that the multiply method in Float has not changed
calculate multiplication
check that Math has not changed
check that the sqrt method in Math has not changed
calculate sqrt
execute a
check that the multiply method in Float has not changed
calculate multiplication
check that the division method in Float has not changed
calculate division

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Improving Performance Using Graal

-b + (Math.sqrt(b**2 - 4*a*c)) / 2*a

execute b
check that the negate method in Float has not changed
calculate negation
execute b
check that the power method in Float has not changed
calculate power
execute a
check that the multiply method in Float has not changed
calculate multiplication
execute c
check that the multiply method in Float has not changed
calculate multiplication
check that Math has not changed
check that the sqrt method in Math has not changed
calculate sqrt
execute a
check that the multiply method in Float has not changed
calculate multiplication
check that the division method in Float has not changed
calculate division

class Float

modified?

module Math

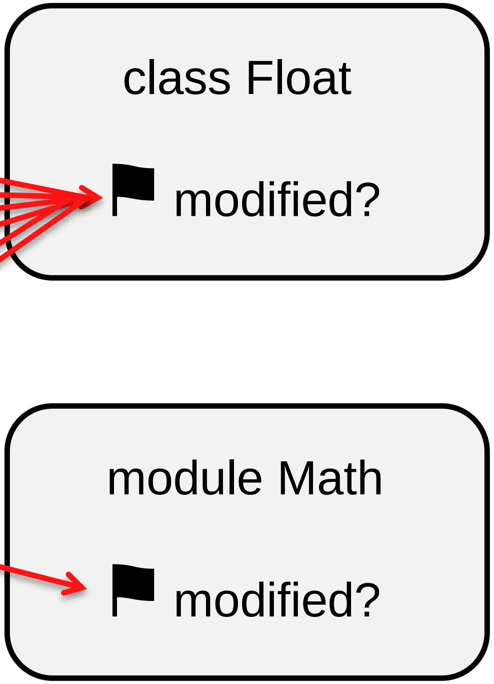
modified?

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Improving Performance Using Graal

-b + (Math.sqrt(b**2 - 4*a*c)) / 2*a

execute b
check that the negate method in Float has not changed
calculate negation
execute b
check that the power method in Float has not changed
calculate power
execute a
check that the multiply method in Float has not changed
calculate multiplication
execute c
check that the multiply method in Float has not changed
calculate multiplication
check that Math has not changed
check that the sqrt method in Math has not changed
calculate sqrt
execute a
check that the multiply method in Float has not changed
calculate multiplication
check that the division method in Float has not changed
calculate division



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Improving Performance Using Graal

-b + (Math.sqrt(b**2 - 4*a*c)) / 2*a

java object InstalledCode

execute b
check that the negate method in Float has not changed
calculate negation
execute b
check that the power method in Float has not changed
calculate power
execute a
check that the multiply method in Float has not changed
calculate multiplication
execute c
check that the multiply method in Float has not changed
calculate multiplication
check that Math has not changed
check that the sqrt method in Math has not changed
calculate sqrt
execute a
check that the multiply method in Float has not changed
calculate multiplication
check that the division method in Float has not changed
calculate division

.invalidate()

class Float

modified?

module Math

modified?

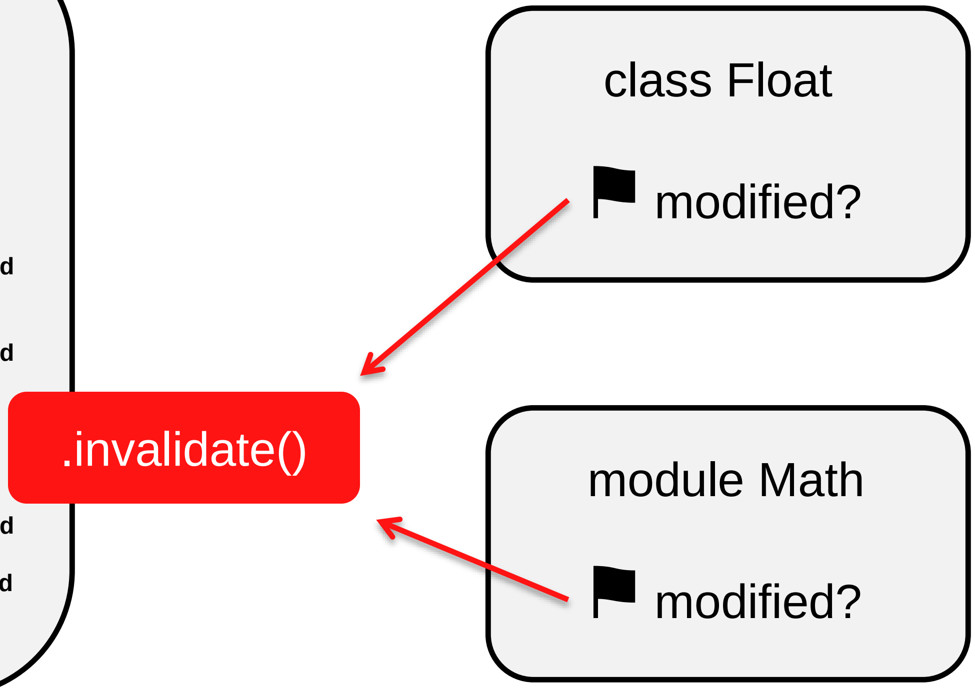
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Improving Performance Using Graal

-b + (Math.sqrt(b**2 - 4*a*c)) / 2*a

java object InstalledCode

execute b
check that the negate method in Float has not changed
calculate negation
execute b
check that the power method in Float has not changed
calculate power
execute a
check that the multiply method in Float has not changed
calculate multiplication
execute c
check that the multiply method in Float has not changed
calculate multiplication
check that Math has not changed
check that the sqrt method in Math has not changed
calculate sqrt
execute a
check that the multiply method in Float has not changed
calculate multiplication
check that the division method in Float has not changed
calculate division



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Improving Performance Using Graal

```
unmodified = new Assumption();

unmodified.check();

unmodified.invalidate();
```


Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Improving Performance Using Graal

-b + (Math.sqrt(b**2 - 4*a*c)) / 2*a

execute b
check that the negate method in Float has not changed
calculate negation
execute b
check that the power method in Float has not changed
calculate power
execute a
check that the multiply method in Float has not changed
calculate multiplication
execute c
check that the multiply method in Float has not changed
calculate multiplication
check that Math has not changed
check that the sqrt method in Math has not changed
calculate sqrt
execute a
check that the multiply method in Float has not changed
calculate multiplication
check that the division method in Float has not changed
calculate division

execute b
calculate negation
execute b
calculate power
execute a
calculate multiplication
execute c
calculate multiplication
calculate sqrt
execute a
calculate multiplication
calculate division

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

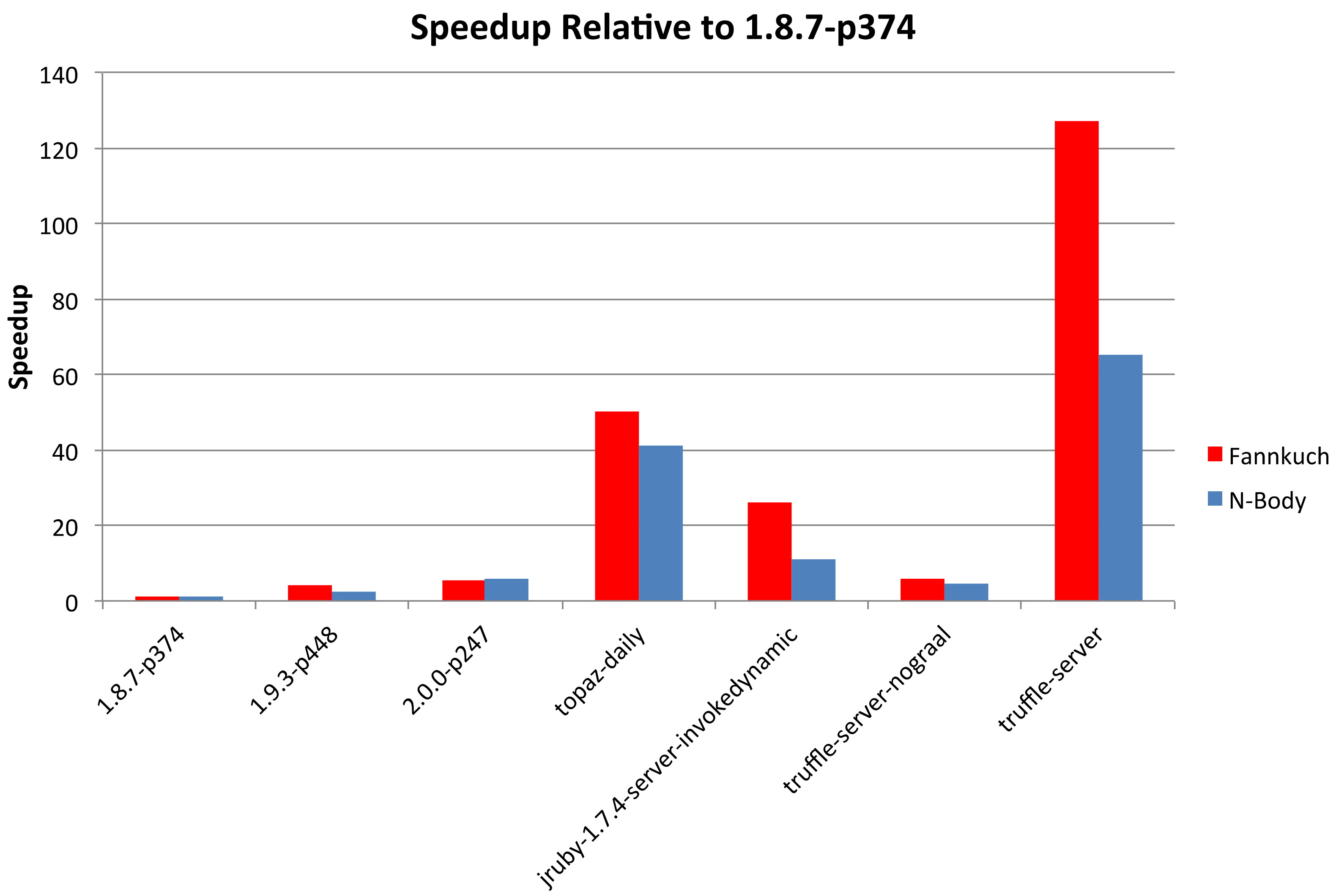
Improving Performance

-b + (Math.sqrt(b**2 - 4*a*c)) / 2*a

execute b
calculate negation
execute b
calculate power
execute a
calculate multiplication
execute c
calculate multiplication
calculate sqrt
execute a
calculate multiplication
calculate division

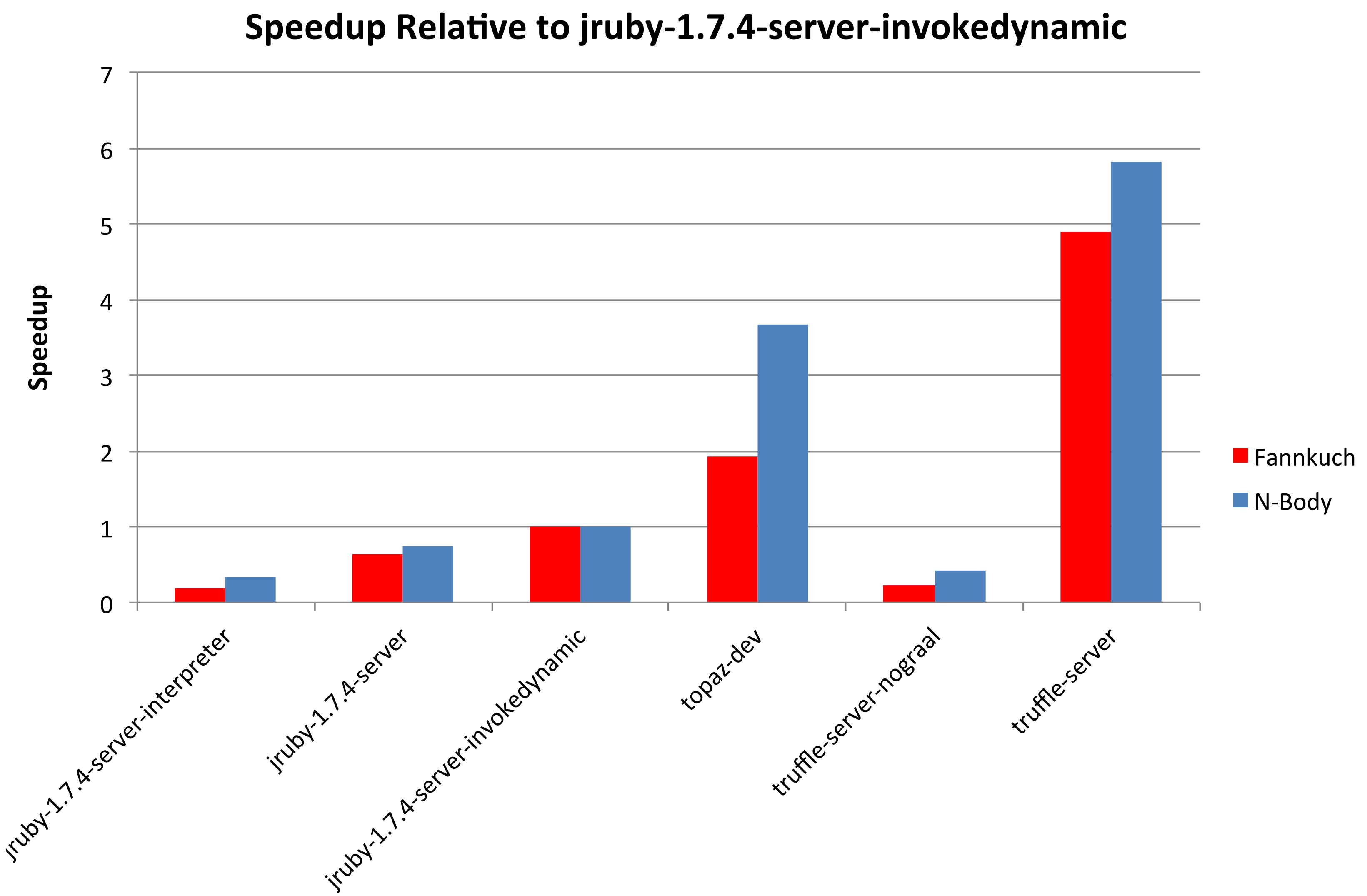
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Peak Performance



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Peak Performance



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Simplicity

- One intern working for five months on the Ruby implementation
- New to Truffle, Graal and Ruby

- Written using Eclipse
- Debugged as a normal Java program using the server compiler
- Run using Graal for testing and performance numbers

- No mention in the implementation of bytecode, classloaders, assembly, system calls, OSR
- One very minor use of Unsafe, one very minor use of reflection

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Acknowledgments

Oracle Labs
Laurent Daynès
Michael Haupt
Peter Kessler
Christos Kotselidis
David Leibs
Roland Schatz
Chris Seaton
Doug Simon
Michael Van De Vanter
Christian Wimmer
Christian Wirth
Mario Wolczko
Thomas Würthinger
Laura Hill (Manager)

Oracle Labs Interns
Danilo Ansaloni
Daniele Bonetta
Shams Imam
Stephen Kell
Helena Kotthaus
Gregor Richards
Rifat Shariyar
Codrut Stancu
Wei Zhang

JKU Linz
Gilles Duboscq
Matthias Grimmer
Christian Häubl
Josef Haider
Christian Humer
Christian Huber
Manuel Rigger
Lukas Stadler
Bernhard Urban
Andreas Wöß

Prof. Hanspeter Mössenböck

Purdue University
Tomas Kalibera
Floreal Morandat
Petr Maj
Prof. Jan Vitek

University of California, Irvine

University of Dortmund

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Hardware and Software

ORACLE

Engineered to Work Together

ORACLE®