
ORACLE®

High-performance Javascript on top of Truffle

Christian Wirth
VM Research Group
Oracle Labs Austria

2014-09-12

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.

Program Agenda

1. Truffle
2. Truffle/JavaScript
3. Other Truffle Languages
4. Research Directions
5. Summary

Truffle

“Write Your Own Language”

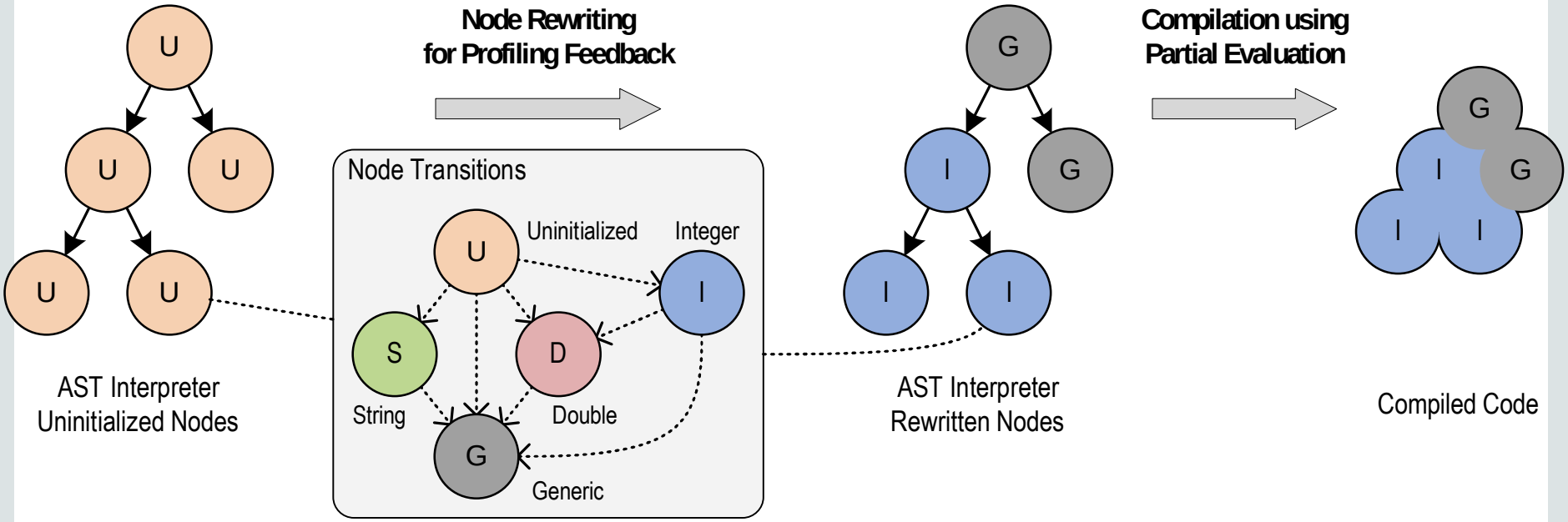
Current situation

- 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

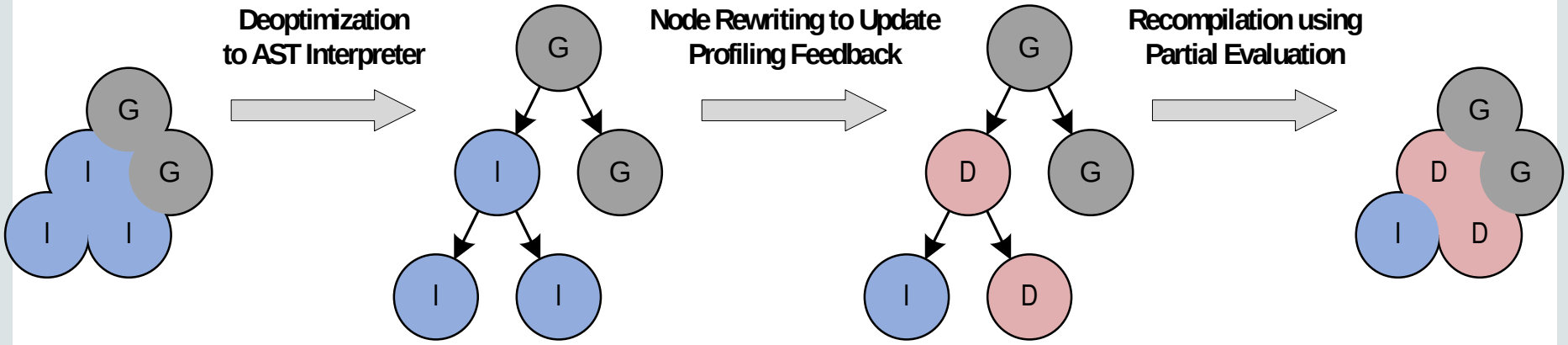
How it should be

- 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

Speculate and Optimize …



… and Deoptimize and Reoptimize!



Truffle DSL

```
@Specialization(rewriteOn=ArithmeticException.class)
int addInt(int a, int b) {
  return Math.addExact(a, b);
}

@Specialization
double addDouble(double a, double b) {
    return a + b;
}

@Generic
Object addGeneric(Frame f, Object a, Object b) {
  // Handling of String omitted for simplicity.
  Number aNum = Runtime.toNumber(f, a);
  Number bNum = Runtime.toNumber(f, b);
  return Double.valueOf(aNum.doubleValue() +
           bNum.doubleValue());
}
```

Truffle API Compiler Directives

- Guards

```
if(condition) {
  // some code that is only valid if condition is true
} else {
  CompilerDirectives.transferToInterpreter();
}
```

- Profiles

```
if (empty.profile(input.isEmpty()) // speculate on condition
```

```
{ branch.enter(); ... } // speculate on branch
```

- Assumptions

```
Assumption assumption = Truffle.getRuntime().createAssumption();
```

```
assumption.check();
// some code that is only valid if assumption is true
```

```
assumption.invalidate();
```

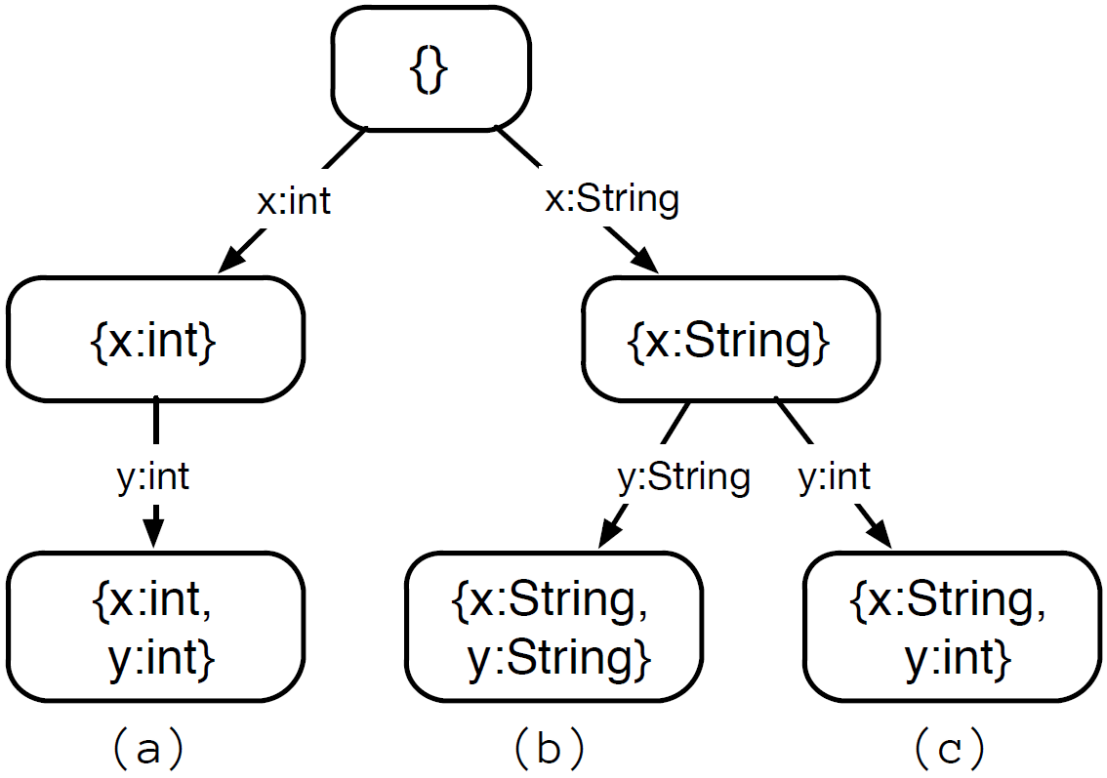
Truffle/JavaScript

Truffle/JavaScript

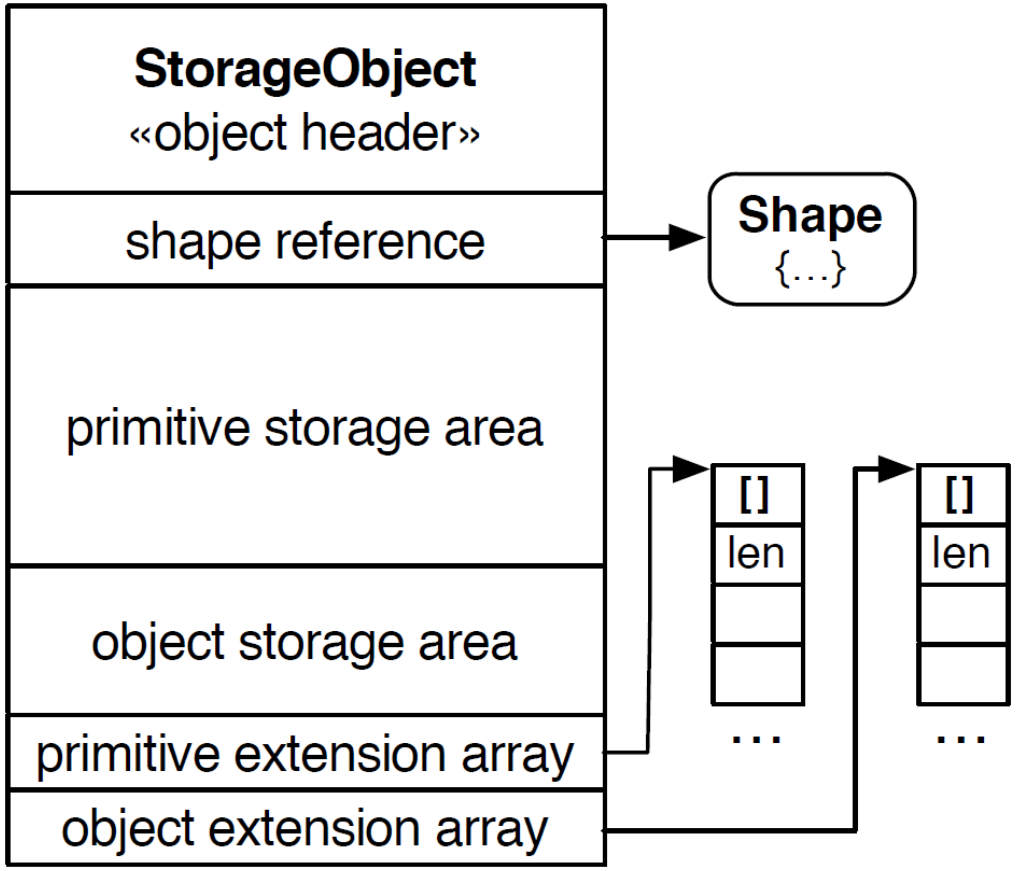
- Oracle Labs, closed source
- A complete language to show Truffle's power
- Supported JavaScript
  - Pass 100% of ECMAScript 5 standard tests
  - Growing support for non-standard extensions
  - Growing support for Node.JS/Avatar.JS applications
- ES5 compliant engine has 60.000 LOC

Object Model

```
var a = {}; a.x = 1; a.y = “foo”;
var b = {}; b.x = “bar”; b.y = “foo”;
var c = {}; c.x = “baz”; c.y = 42;
```

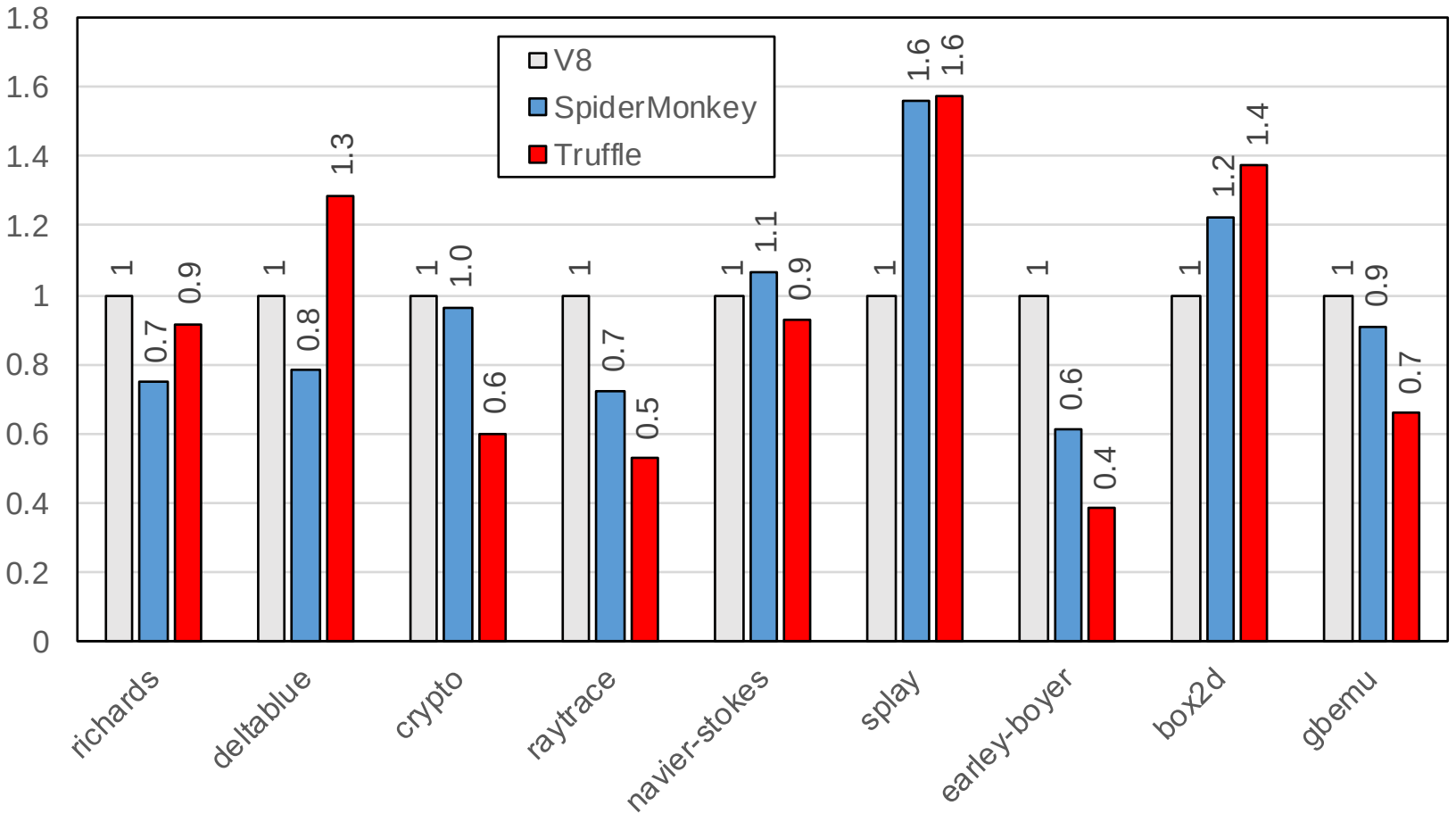


Object Storage Model



Wöß, Wirth, Bonetta, Seaton, Humer, and Mössenböck. **An object storage model for the truffle language implementation framework** (PPPJ '14)

Performance



Other Truffle Languages

Other Truffle Languages

C

- Truffle/C: Oracle Labs, JKU Linz
- Closed source

Ruby

- Oracle Labs, experimental part of JRuby
- Open source: https://github.com/jruby/jruby

R

- FastR: Oracle Labs, JKU Linz, Purdue University
- Open source: https://bitbucket.org/allr/fastr

Python

- ZipPy: UC Irvine
- Open source: https://bitbucket.org/ssllab/zippy/

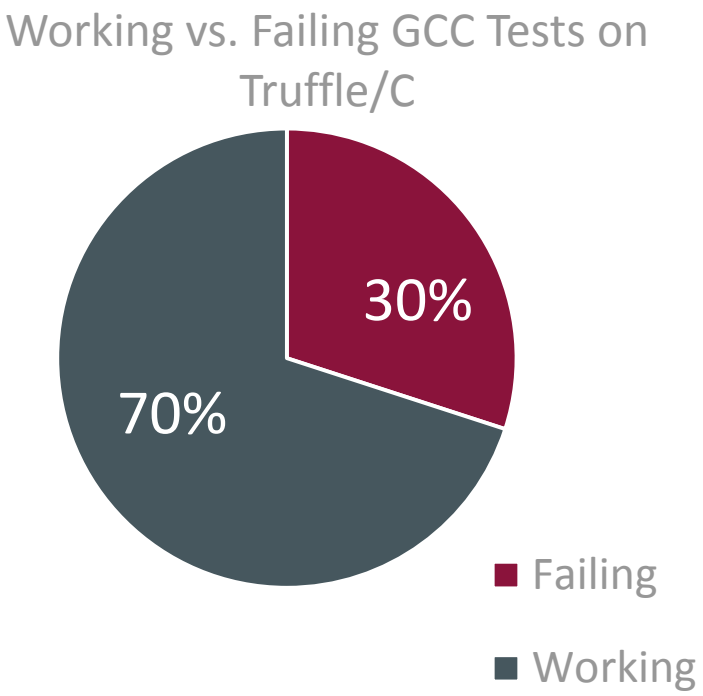
SOM (Smalltalk)

- Stefan Marr, INRIA
- Open source: https://github.com/smarr/TruffleSOM

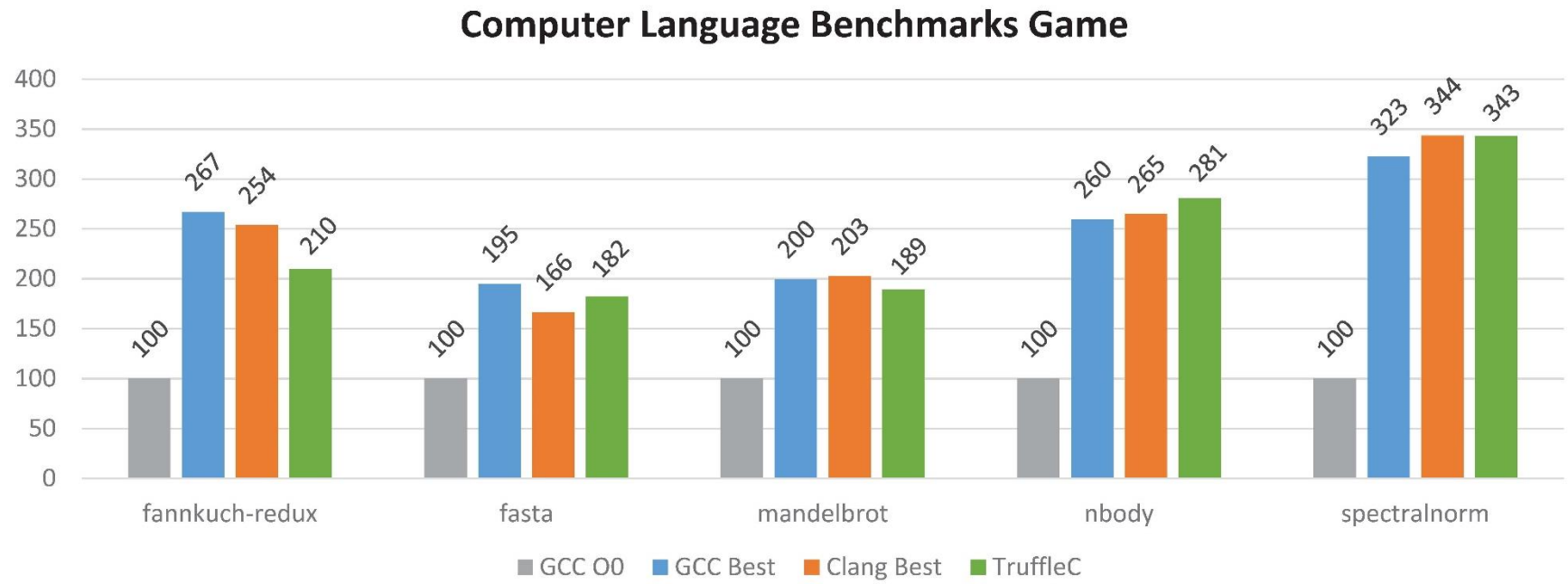
Truffle/C

How complete is Truffle/C?

- Close to ANSI C90
- GCC test suite: torture tests



Truffle/C Performance



Higher is better

100% Baseline is GCC –O0

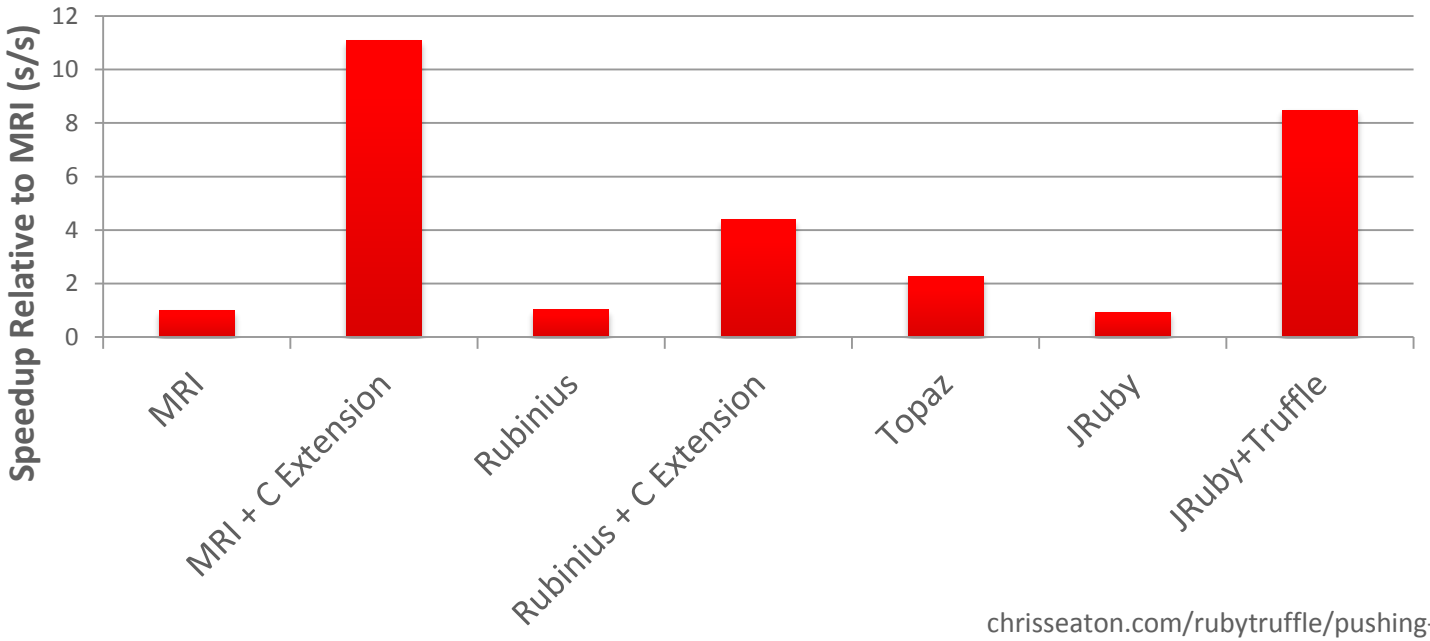
GCC = max(GCC O1, O2, O3)

Clang = max(Clang O1, O2, O3)

JRuby+Truffle

- Truffle is an experimental backend in JRuby
- Working to be 100% compliant with MRI
  - All language features implemented
  - Pure engineering effort to support whole library
- Better support for some features than JRuby
  - Full **ObjectSpace** and **set_trace_func** are always enabled
  - Aliasing of methods like **binding** and **eval**
  - Pure Ruby call stacks with all Ruby local variables

Performance on chunky_png and psd.rb



chrisseaton.com/rubytruffle/pushing-pixels

- Real code, unmodified from the original gems
- Includes diverse Ruby code such as arithmetics, method calls, dynamic features like #send, …
- Geometric mean across all benchmarks

Research Directions

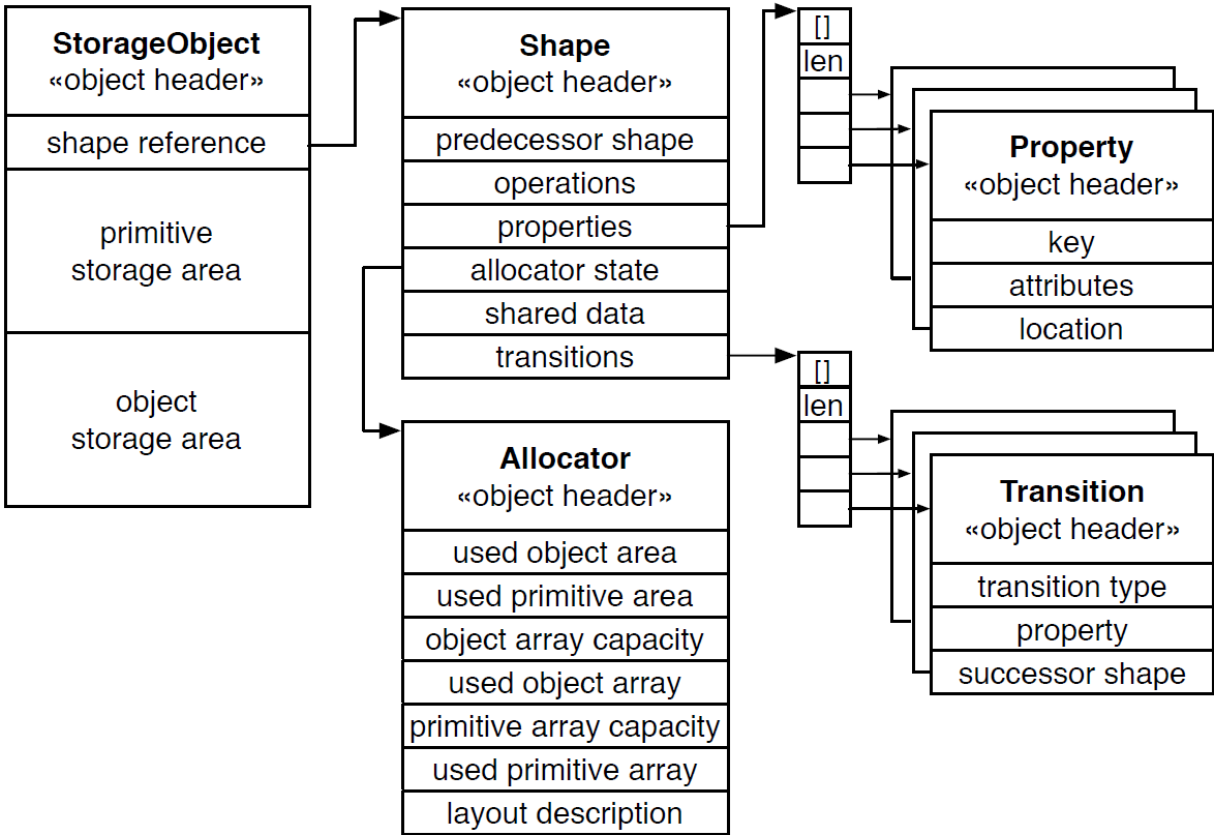
Research Contributions

- Würthinger, Wimmer, Wöß, Stadler, Duboscq, Humer, Richards, Simon, and Wolczko. 2013. One **VM to rule them all**. In *Proceedings of the 2013 ACM international symposium on New ideas, new paradigms, and reflections on programming & software* (Onward! '13).
- Würthinger, Wöß, Stadler, Duboscq, Simon, and Wimmer. 2012. **Self-optimizing AST interpreters**. In *Proceedings of the 8th symposium on Dynamic languages* (DLS '12).
- Grimmer, Rigger, Schatz, Stadler, and Mössenböck. 2014. **TruffleC: dynamic execution of C on a Java virtual machine**. In *Proceedings of the 2014 International Conference on Principles and Practices of Programming on the Java platform: Virtual machines, Languages, and Tools* (PPPJ '14).
- Grimmer, Würthinger, Wöß, and Mössenböck. 2014. **An efficient approach for accessing C data structures from JavaScript**. In *Proceedings of the 9th International Workshop on Implementation, Compilation, Optimization of Object-Oriented Languages, Programs and Systems PLE* (ICOOOLPS '14).
- Seaton, Van De Vanter, and Haupt. 2014. **Debugging at Full Speed**. In *Proceedings of the Workshop on Dynamic Languages and Applications* (Dyla'14).
- Wöß, Wirth, Bonetta, Seaton, Humer, and Mössenböck. 2014. **An object storage model for the truffle language implementation framework**. In *Proceedings of the 2014 International Conference on Principles and Practices of Programming on the Java platform: Virtual machines, Languages, and Tools* (PPPJ '14).
- Humer, Wimmer, Wirth, Wöß, and Würthinger. 2014. **A domain-specific language for building self-optimizing AST interpreters**. In *Proceedings of the 2014 International Conference on Generative Programming: Concepts and Experiences* (GPCE 2014)

… and more by our research partners

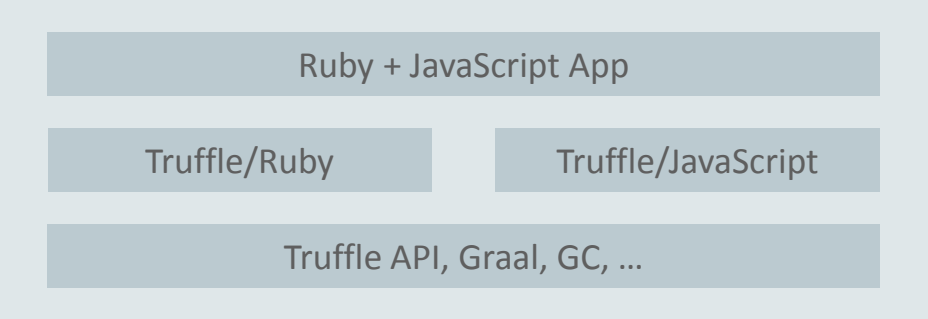
ORACLE

Object Storage Model for Truffle



Wöß, Wirth, Bonetta, Seaton, Humer, and Mössenböck. **An object storage model for the truffle language implementation framework** (PPPJ '14)

Language Interoperability



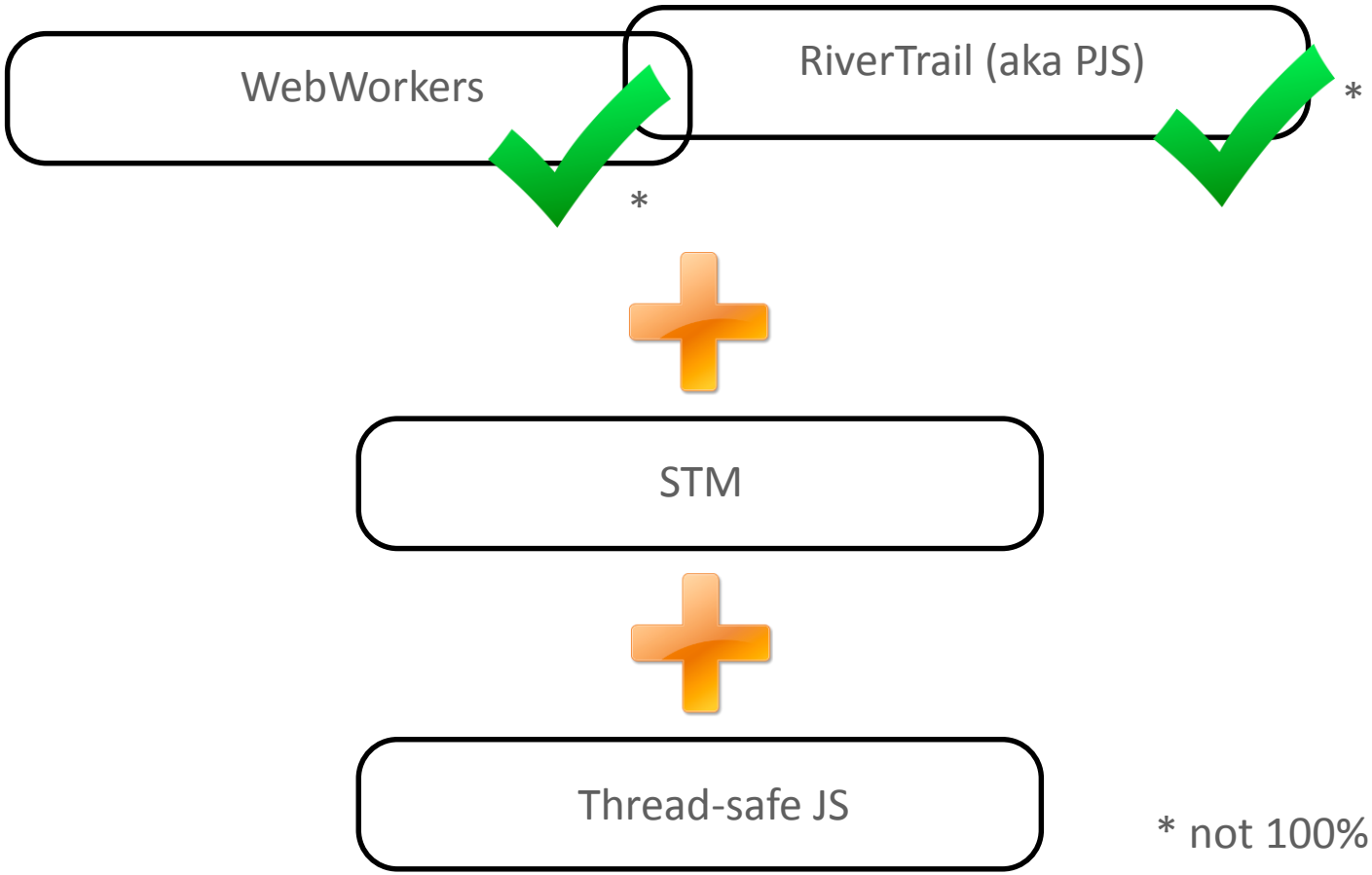
One Modular VM to Rule Them All

The shared optimization infrastructure of Truffle allows simple and fast language integration

- Our claim: performance of combined language execution is as good as the individual VMs
  - The compiler fuses language checks into existing inline cache checks
  - Approach performs cross-language method inlining
  - Approach exchanges primitive values between languages without boxing or conversions

Grimmer, Würthinger, Wöß, and Mössenböck: **An efficient approach for accessing C data structures from JavaScript** (ICOOOLPS '14)

Parallel JavaScript



* not 100% complete

Substrate VM

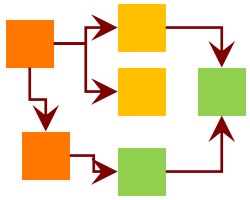
Static Analysis

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 compilation or Java class loading

Summary

Summary

- Language implementation framework
- High-performance language engine
- Open-source engines for several languages available
- Ongoing research in several directions

Your language?

http://openjdk.java.net/projects/graal/

graal-dev@openjdk.java.net

```
$ hg clone http://hg.openjdk.java.net/graal/graal
$ cd graal
$ ./mx --vm server build
$ ./mx ideinit
$ ./mx --vm server unittest SumTest
```

Truffle API Resources

https://wiki.openjdk.java.net/display/Graal/Truffle+FAQ+and+Guidelines

Truffle API License: GPLv2 with Classpath Exception

Hardware and Software
Engineered to Work Together


ORACLE®