Language Composition with RPython
PyConUK 2015

Edd Barrett
Lukas Diekmann
Laurence Tratt
Carl Friedrich Bolz

Software Development Team
September 19, 2015
Roadmap

- Language composition background
- Challenges
- Our approach
- Concrete example: PHP + Python
Language Composition

“The ability to write a computer program in a mixture of programming languages.”
Why Compose Languages?

- Different parts of a program expressed best with different langs
- Language migration
What about FFIs?

```python
import cffi
```
Rethinking the FFI

code.py

| call |

| code.c |

return
Rethinking the FFI

code.py
any language

---

code.c
any language

---
call

---
return

any language

---

6/44 HTTP://SOFT-DEV.ORG/
Rethinking the FFI

my_composed_program
From different languages, mix:

- modules
- functions
- methods
- expressions
- scopes
Users won’t adopt composition if it slows their programs down
We want language compositions which are: 

FINE-GRAINED and FAST

How?
Breaking it Down

PL X

PL Y

PL Z

10/44 HTTP://SOFT-DEV.ORG/
Composing Syntax

PL X
<grammar>
expr ::= ...
term ::= ...
    | ...
func ::= ...
PL Y
<grammar>
expr ::= ...
term ::= ...
    | ...
func ::= ...
PL Z
<grammar>
expr ::= ...
term ::= ...
    | ...
func ::= ...

= ∪

Easy?
Composing Syntax

- LR → Possibly undefined
- PEG → Shadows
- GLR → Ambiguous
Syntax Directed Editing

public class Say extends <none> implements <none> {

    private String textchanged;
    <<<properties>>>
    <<<initializer>>>
    public Say(String text) {
        <<<no statements>>>
    }

    <<<methods>>>

    <<<nested classifiers>>>
}

Poor editing experience
Syntax Composition
Composing Runtimes

RT X

RT Y

RT Z

Easy?
Approaches to Interpreter Composition

Edd Barrett, Carl Friedrich Bolz, Laurence Tratt

Software Development Team, Informatics, King's College London, UK

Abstract

In this paper, we compose six different Python and Prolog VMs into 4 pairwise compositions: one using C interpreters; one running on the JVM; one using meta-tracing interpreters; and one using a C interpreter and a meta-tracing interpreter. We show that programs that cross the language barrier frequently execute faster in a meta-tracing composition, and that meta-tracing imposes a significantly lower overhead on composed programs relative to mono-language programs.

Keywords: Language composition, Virtual machines

1. Overview

Programming language composition aims to allow the mixing of programming languages in a fine-grained manner. This vision brings many challenging problems, from the interaction of language semantics to performance. In this paper, we investigate the runtime performance of composed programs in high-level languages. We start from the assumption that execution of such programs is most likely to be through composing language implementations that use interpreters.
Runtime composition

PL X

Interpreter

PL Y

Interpreter

C/C++
Runtime composition

PL X

Interpreter

JIT Compiler

PL Y

Interpreter

C/C++
Runtime composition

Too much engineering
Runtime composition

PL X
Interpreter

PL Y
Interpreter

JVM/CLR
JIT Compiler

17/44 HTTP://SOFT-DEV.ORG/
Poor performance for dynamic languages
Runtime Composition

Syntactic Composition
Our Approach

Language Boxes + Meta-tracing
Language Boxes

- Borrows ideas from SDE
- Palatable editing experience
- Simple and practical way to compose grammars
Begin writing Java code

for (string s :
for (string s :
Open SQL language box
Language Boxes: E.g. Java + SQL

for (string s : SELECT * FROM tbl WHERE
for (string s : SELECT * FROM tbl WHERE name = this.name;) {
  
Java code
Syntactic Composition
Meta-tracing

Interpreter → Meta-tracing

Meta-tracing → Interpreter and Tracing JIT
Meta-tracing

Python

Interpreter

RPython

Python

Interpreter

Tracing JIT

pypy
How Does this Apply to VM Composition?

Little Engineering + High Performance
Runtime Composition
A Concrete Example: PyHyp
PyHyp

Language box editor
FFI-like Features

- Import Python modules into PHP
- Calling Python functions and methods from PHP
- Calling PHP functions and methods from Python
- Automatic type “conversion”
“Advanced Features”

- Foreign functions in the same file
  - And arbitrarily nested

- Python expressions in PHP

- “Embedding” Python methods inside PHP classes
  - And integration with PHP access modifiers

- Adds support for references to Python

- Cross-language scoping

- Mixed-frames in backtraces new!
Demo
Implementing desired behaviour: relatively easy

Deciding the correct behaviours: hard

Compromises sometimes must be made
Example: Collection types across languages
Array/Dict/List Conversions

PHP  Language Threshold  Python

int  int
str  str
obj  obj
"adapted"
Array/Dict/List Conversions

PHP ↔ Language Threshold ↔ Python

- **array**
  - list (integer keys)
- **array**
  - dict (mixed keys)
- **list**
- **dict**

33/44 HTTP://SOFT-DEV.ORG/
Array/Dict/List Conversions

PHP  

Language Threshold  

Python  

array  
int keys  

list  
array  

array  
mixed keys  

dict  
array  

33/44 HTTP://SOFT-DEV.ORG/
Array/Dict/List Conversions

PHP ⟷ Language Threshold ⟷ Python

$a = \text{array} \quad \Rightarrow \quad \text{list}

\text{int keys}

\text{name}

$a["x"] = 4

mixed keys
**Array/Dict/List Conversions**

Language Threshold

PHP

Python

$a=$array

int keys

$a["x"] = 4

array

mixed keys

list

array

Inconsistent list!
Array/Dict/List Conversions

PHP ← Language Threshold → Python

```
array
12
dict
array
12
list
array
```

```
array
→
dict
array
```

```
array
←
list
array
```

```
array
←
list
array
```

```
array
as_list()
dict
```

34 / 44
http://soft-dev.org/
Performance Evaluation
Composed programs should perform “close” to equivalent mono-language programs

Aim for between 1-2x slower. 3x is too slow
Benchmark “Variants”

Variant 1
PHP

Variant 2
Python

Variant 3
PHP + Python

Variant 4
Python + PHP

37 / 44 HTTP://SOFT-DEV.ORG/
Benchmark “Variants”

Variants:
- Variant 1: PHP
- Variant 2: Python
- Variant 3: PHP + Python
- Variant 4: Python + PHP

TODO: XXX

Source: http://soft-dev.org/
Benchmarks

- deltablue
- fannkuch
- mandel
- richards
- instchain
- l1a0r
- l1a1r
- lists
- ref_swap

- return_simple
- scopes
- smallfunc
- sum
- sum_meth
- sum_meth_attr
- total_list
- walk_list
### Microbenchmarks: Relative to PyHyp Variant3

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>CPython</th>
<th>HHVM</th>
<th>HippyVM</th>
<th>PyHyp\textsubscript{m}</th>
<th>PyPy</th>
<th>Zend</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>instchain</strong></td>
<td>22.172</td>
<td>±0.0859</td>
<td>6.209</td>
<td>0.969</td>
<td>0.967</td>
<td>0.477</td>
</tr>
<tr>
<td><strong>l1a0r</strong></td>
<td>71.633</td>
<td>±1.4869</td>
<td>3.770</td>
<td>1.230</td>
<td>1.230</td>
<td>1.231</td>
</tr>
<tr>
<td><strong>l1a1r</strong></td>
<td>76.171</td>
<td>±0.1207</td>
<td>3.000</td>
<td>1.285</td>
<td>1.285</td>
<td>1.144</td>
</tr>
<tr>
<td><strong>lists</strong></td>
<td>7.485</td>
<td>±0.0227</td>
<td>0.879</td>
<td>0.966</td>
<td>0.977</td>
<td>0.520</td>
</tr>
<tr>
<td><strong>ref_swap</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6.911</td>
<td>±0.0054</td>
</tr>
<tr>
<td><strong>return_simple</strong></td>
<td>108.576</td>
<td>±0.2690</td>
<td>6.915</td>
<td>1.000</td>
<td>1.000</td>
<td>0.889</td>
</tr>
<tr>
<td><strong>scopes</strong></td>
<td>123.284</td>
<td>±1.5081</td>
<td>14.969</td>
<td>4.528</td>
<td>4.512</td>
<td>1.000</td>
</tr>
<tr>
<td><strong>smallfunc</strong></td>
<td>184.778</td>
<td>±0.3071</td>
<td>12.818</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td><strong>sum</strong></td>
<td>299.582</td>
<td>±0.3659</td>
<td>19.083</td>
<td>1.000</td>
<td>1.000</td>
<td>0.874</td>
</tr>
<tr>
<td><strong>sum_meth</strong></td>
<td>328.894</td>
<td>±1.2870</td>
<td>23.714</td>
<td>0.998</td>
<td>0.999</td>
<td>0.873</td>
</tr>
<tr>
<td><strong>sum_meth_attr</strong></td>
<td>127.800</td>
<td>±0.1907</td>
<td>17.819</td>
<td>1.001</td>
<td>1.116</td>
<td>0.925</td>
</tr>
<tr>
<td><strong>total_list</strong></td>
<td>14.266</td>
<td>±0.0248</td>
<td>2.080</td>
<td>0.695</td>
<td>0.696</td>
<td>0.510</td>
</tr>
<tr>
<td><strong>walk_list</strong></td>
<td>4.869</td>
<td>±0.0340</td>
<td>0.373</td>
<td>0.773</td>
<td>0.774</td>
<td>1.099</td>
</tr>
</tbody>
</table>
## Larger Benchmarks: Relative to PyHyp Variant3

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>CPython</th>
<th>HHVM</th>
<th>HippyVM</th>
<th>PyHyp&lt;sub&gt;m&lt;/sub&gt;</th>
<th>PyPy</th>
<th>Zend</th>
</tr>
</thead>
<tbody>
<tr>
<td>deltablue</td>
<td>19.199</td>
<td>860.108</td>
<td>4.739</td>
<td>4.888</td>
<td>0.405</td>
<td>181.209</td>
</tr>
<tr>
<td></td>
<td>±0.6900</td>
<td>±31.1392</td>
<td>±0.1684</td>
<td>±0.1766</td>
<td>±0.0151</td>
<td>±6.7871</td>
</tr>
<tr>
<td>fannkuch</td>
<td>18.616</td>
<td>3.212</td>
<td>1.869</td>
<td>1.879</td>
<td>1.009</td>
<td>14.998</td>
</tr>
<tr>
<td></td>
<td>±0.0362</td>
<td>±0.0133</td>
<td>±0.0034</td>
<td>±0.0024</td>
<td>±0.0046</td>
<td>±0.1032</td>
</tr>
<tr>
<td>mandel</td>
<td>0.883</td>
<td>1.013</td>
<td>1.003</td>
<td>1.003</td>
<td>8.290</td>
<td></td>
</tr>
<tr>
<td></td>
<td>±0.0006</td>
<td>±0.0089</td>
<td>±0.0011</td>
<td></td>
<td>±0.0623</td>
<td></td>
</tr>
<tr>
<td>Richards</td>
<td>28.291</td>
<td>12.726</td>
<td>0.745</td>
<td>0.766</td>
<td>0.531</td>
<td>27.081</td>
</tr>
<tr>
<td></td>
<td>±0.1091</td>
<td>±0.1296</td>
<td>±0.0036</td>
<td>±0.0044</td>
<td>±0.0026</td>
<td>±0.1445</td>
</tr>
</tbody>
</table>
## Overall: Relative to PyHyp Variant3

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>CPython</th>
<th>HHVM</th>
<th>HippyVM</th>
<th>PyHyp&lt;sub&gt;m&lt;/sub&gt;</th>
<th>PyPy</th>
<th>Zend</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geometric Mean</td>
<td>48.575 ± 0.1493</td>
<td>6.698 ± 0.0188</td>
<td>1.206 ± 0.0032</td>
<td>1.218 ± 0.0035</td>
<td>0.785 ± 0.0024</td>
<td>56.521 ± 0.1833</td>
</tr>
</tbody>
</table>
## Overall: Relative to PyHyp Variant3

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>CPython</th>
<th>HHVM</th>
<th>HippyVM</th>
<th>PyHyp(_m)</th>
<th>PyPy</th>
<th>Zend</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geometric Mean</td>
<td>48.575</td>
<td>6.698</td>
<td>1.206</td>
<td>1.218</td>
<td>0.785</td>
<td>56.521</td>
</tr>
<tr>
<td></td>
<td>±0.1493</td>
<td>±0.0188</td>
<td>±0.0032</td>
<td>±0.0035</td>
<td>±0.0024</td>
<td>±0.1833</td>
</tr>
</tbody>
</table>
Conclusions

- **Language boxes:**
  - Practical syntax composition
  - Decent editor experience

- **Meta-tracing:**
  - Compositions with relatively little effort
  - Overall good performance

- Implementing x-lang behaviours is easy
- Designing x-lang behaviours is hard
Future Work

- Implement and measure benchmark variant 4
- Cross-language debugger
- Compositions with >2 languages involved
- Statically typed languages
Thanks

Hey Kurt,
I recently came up with this:

```
\text{Eq}(G_{\mu\nu} + \Lambda g_{\mu\nu}, (B^2 \mp \text{G}*c^2)^4) (T_{\mu\nu})^
```

What do you think?

Albert