Language integration and migration

Edd Barrett
Carl Friedrich Bolz
Lukas Diekmann
Laurence Tratt
Naveneetha Krishnan Vasudevan

Software Development Team
2014-09-11

KING'S
College
LONDON
What to expect from this talk

A

B
What to expect from this talk

\[ A \cup B \]
What to expect from this talk

Python ∪ Prolog
What to expect from this talk

Python ∪ PHP
Our problem
Our problem

We want better programming languages
We want better programming languages

But better always seems to end up bigger
Underlying language composition challenges
Underlying language composition challenges

Python

PHP

Bridge
Underlying language composition challenges

Python
syntax
runtime

PHP
syntax
runtime

Bridge
syntax
runtime
Underlying language composition challenges

Language boxes

Syntax
Runtime

Python

Syntax
Runtime

PHP

Syntax
Runtime

Bridge
Underlying language composition challenges

Syntax

Python
Runtime

PHP
Syntax
Runtime

Language boxes

Syntax

Bridge
Runtime

Composed meta-tracing VMs
<grammar>
expr ::= ...
term ::= ...
    | ...
    | ...
func ::= ...
<program>
for (j : js) {
    doStuff();
}
.
Syntax composition

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

PL Y
<program>
for (j : js) {
    doStuff();
}
.
.
.
Parser
Syntax composition

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

PL Y
<program>
for (j : js) {
    doStuff();
}
.
.
.
Parser
Parse Tree

5/22 HTTP://SOFT-DEV.ORg/
The only choice?
The only choice?

SDE
Challenge:
SDE’s power +
a text editor feel?
Runtime composition
Runtime composition

PL X
Interpreter

PL Y
Interpreter

C/C++
Runtime composition

PL X
- JIT Compiler
- Interpreter

PL Y
- JIT Compiler
- Interpreter

C/C++
Runtime composition

PL X

Interpreter

JVM/CLR

Interpreter

JIT Compiler

PL Y

Interpreter

JIT Compiler
Warning: draft numbers ahead
### Absolute timing comparison

<table>
<thead>
<tr>
<th>VM</th>
<th>Benchmark</th>
<th>Python</th>
<th>Prolog</th>
<th>Python → Prolog</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0.125s ± 0.006</td>
<td>0.257s ± 0.001</td>
<td>28.893s ± 0.175</td>
</tr>
<tr>
<td>CPython-SWI</td>
<td>SmallFunc</td>
<td>2.924s ± 0.215</td>
<td>7.352s ± 0.037</td>
<td>9.310s ± 0.065</td>
</tr>
<tr>
<td></td>
<td>Loop1Arg0Result</td>
<td>4.184s ± 0.028</td>
<td>18.890s ± 0.082</td>
<td>20.865s ± 0.050</td>
</tr>
<tr>
<td></td>
<td>Loop1Arg1Result</td>
<td>7.531s ± 0.065</td>
<td>18.643s ± 0.159</td>
<td>667.682s ± 5.594</td>
</tr>
<tr>
<td></td>
<td>NondetLoop1Arg1Result</td>
<td>264.415s ± 1.815</td>
<td>48.819s ± 0.208</td>
<td>2185.150s ± 14.251</td>
</tr>
<tr>
<td></td>
<td>TermConstruction</td>
<td>9.374s ± 0.046</td>
<td>25.148s ± 0.182</td>
<td>2207.304s ± 12.344</td>
</tr>
<tr>
<td>Unipyccation</td>
<td>SmallFunc</td>
<td>0.001s ± 0.000</td>
<td>0.006s ± 0.001</td>
<td>0.001s ± 0.000</td>
</tr>
<tr>
<td></td>
<td>Loop1Arg0Result</td>
<td>0.085s ± 0.000</td>
<td>0.086s ± 0.000</td>
<td>0.087s ± 0.000</td>
</tr>
<tr>
<td></td>
<td>Loop1Arg1Result</td>
<td>0.112s ± 0.000</td>
<td>0.114s ± 0.000</td>
<td>0.115s ± 0.000</td>
</tr>
<tr>
<td></td>
<td>NondetLoop1Arg1Result</td>
<td>0.500s ± 0.002</td>
<td>0.548s ± 0.064</td>
<td>2.674s ± 0.010</td>
</tr>
<tr>
<td></td>
<td>TermConstruction</td>
<td>6.053s ± 0.218</td>
<td>2.444s ± 0.002</td>
<td>36.069s ± 0.171</td>
</tr>
<tr>
<td></td>
<td>Lists</td>
<td>0.845s ± 0.002</td>
<td>1.416s ± 0.003</td>
<td>5.056s ± 0.026</td>
</tr>
<tr>
<td>Jython-tuProlog</td>
<td>SmallFunc</td>
<td>0.088s ± 0.002</td>
<td>3.050s ± 0.036</td>
<td>52.294s ± 0.371</td>
</tr>
<tr>
<td></td>
<td>Loop1Arg0Result</td>
<td>1.078s ± 0.007</td>
<td>206.590s ± 2.884</td>
<td>199.963s ± 1.784</td>
</tr>
<tr>
<td></td>
<td>Loop1Arg1Result</td>
<td>2.145s ± 0.175</td>
<td>293.311s ± 4.270</td>
<td>294.781s ± 4.746</td>
</tr>
<tr>
<td></td>
<td>NondetLoop1Arg1Result</td>
<td>7.939s ± 0.341</td>
<td>timeout</td>
<td>timeout</td>
</tr>
<tr>
<td></td>
<td>TermConstruction</td>
<td>timeout</td>
<td>timeout</td>
<td>timeout</td>
</tr>
<tr>
<td></td>
<td>Lists</td>
<td>timeout</td>
<td>timeout</td>
<td>timeout</td>
</tr>
</tbody>
</table>
## Relative timing comparison

<table>
<thead>
<tr>
<th>VM</th>
<th>Benchmark</th>
<th>Python $\rightarrow$ Prolog</th>
<th>Python $\rightarrow$ Prolog</th>
<th>Python $\rightarrow$ Prolog</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Python</td>
<td>Prolog</td>
<td>Unipyration</td>
</tr>
<tr>
<td>CPython-SWI</td>
<td>SmallFunc</td>
<td>$231.770 \times \pm 0.154$</td>
<td>$112.567 \times \pm 0.934$</td>
<td>$27821.079 \times \pm 1896.725$</td>
</tr>
<tr>
<td></td>
<td>Loop1Arg0Result</td>
<td>$3.184 \times \pm 0.232$</td>
<td>$1.266 \times \pm 0.011$</td>
<td>$107.591 \times \pm 0.779$</td>
</tr>
<tr>
<td></td>
<td>Loop1Arg1Result</td>
<td>$4.987 \times \pm 0.039$</td>
<td>$1.105 \times \pm 0.006$</td>
<td>$181.899 \times \pm 0.444$</td>
</tr>
<tr>
<td></td>
<td>NondetLoop1Arg1Result</td>
<td>$88.654 \times \pm 1.026$</td>
<td>$35.814 \times \pm 0.389$</td>
<td>$249.737 \times \pm 2.244$</td>
</tr>
<tr>
<td></td>
<td>TermConstruction</td>
<td>$8.264 \times \pm 0.081$</td>
<td>$44.760 \times \pm 0.348$</td>
<td>$60.583 \times \pm 0.487$</td>
</tr>
<tr>
<td></td>
<td>Lists</td>
<td>$235.459 \times \pm 1.742$</td>
<td>$87.772 \times \pm 0.789$</td>
<td>$436.609 \times \pm 3.494$</td>
</tr>
<tr>
<td>Unipyration</td>
<td>SmallFunc</td>
<td>$1.295 \times \pm 0.086$</td>
<td>$0.182 \times \pm 0.036$</td>
<td>$1.000 \times \pm 0.000$</td>
</tr>
<tr>
<td></td>
<td>Loop1Arg0Result</td>
<td>$1.020 \times \pm 0.001$</td>
<td>$1.012 \times \pm 0.002$</td>
<td>$1.000 \times \pm 0.000$</td>
</tr>
<tr>
<td></td>
<td>Loop1Arg1Result</td>
<td>$1.025 \times \pm 0.002$</td>
<td>$1.002 \times \pm 0.002$</td>
<td>$1.000 \times \pm 0.000$</td>
</tr>
<tr>
<td></td>
<td>NondetLoop1Arg1Result</td>
<td>$5.349 \times \pm 0.035$</td>
<td>$4.879 \times \pm 0.0631$</td>
<td>$1.000 \times \pm 0.000$</td>
</tr>
<tr>
<td></td>
<td>TermConstruction</td>
<td>$5.959 \times \pm 0.224$</td>
<td>$14.756 \times \pm 0.069$</td>
<td>$1.000 \times \pm 0.000$</td>
</tr>
<tr>
<td></td>
<td>Lists</td>
<td>$5.982 \times \pm 0.034$</td>
<td>$3.569 \times \pm 0.019$</td>
<td>$1.000 \times \pm 0.000$</td>
</tr>
<tr>
<td>Jython-tuProlog</td>
<td>SmallFunc</td>
<td>$592.904 \times \pm 14.602$</td>
<td>$17.143 \times \pm 0.259$</td>
<td>$50354.204 \times \pm 3330.993$</td>
</tr>
<tr>
<td></td>
<td>Loop1Arg0Result</td>
<td>$185.460 \times \pm 2.182$</td>
<td>$0.968 \times \pm 0.017$</td>
<td>$2310.844 \times \pm 21.996$</td>
</tr>
<tr>
<td></td>
<td>Loop1Arg1Result</td>
<td>$137.427 \times \pm 11.805$</td>
<td>$1.005 \times \pm 0.022$</td>
<td>$2569.873 \times \pm 41.331$</td>
</tr>
<tr>
<td></td>
<td>NondetLoop1Arg1Result</td>
<td>timeout</td>
<td>timeout</td>
<td>timeout</td>
</tr>
<tr>
<td></td>
<td>TermConstruction</td>
<td>timeout</td>
<td>timeout</td>
<td>timeout</td>
</tr>
<tr>
<td></td>
<td>Lists</td>
<td>timeout</td>
<td>timeout</td>
<td>timeout</td>
</tr>
</tbody>
</table>
PHP / Python bridge demo
Warning: even draftier numbers ahead!
## Composed Richards vs. other VMs

<table>
<thead>
<tr>
<th>Type</th>
<th>VM</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mono</td>
<td>PyPy 2.4.0</td>
<td>0.370 ± 0.000</td>
</tr>
<tr>
<td></td>
<td>Hippy</td>
<td>0.553 ± 0.008</td>
</tr>
<tr>
<td></td>
<td>Bridge</td>
<td>0.556 ± 0.006</td>
</tr>
<tr>
<td></td>
<td>HHVM 3.2.0</td>
<td>5.353 ± 0.262</td>
</tr>
<tr>
<td></td>
<td>ZEND 5.4.4</td>
<td>10.406 ± 0.106</td>
</tr>
<tr>
<td>Type</td>
<td>VM</td>
<td>Value</td>
</tr>
<tr>
<td>------------</td>
<td>---------------</td>
<td>-----------</td>
</tr>
<tr>
<td>Mono</td>
<td>PyPy 2.4.0</td>
<td>0.370 ± 0.000</td>
</tr>
<tr>
<td></td>
<td>Hippy</td>
<td>0.553 ± 0.008</td>
</tr>
<tr>
<td></td>
<td>Bridge</td>
<td>0.556 ± 0.006</td>
</tr>
<tr>
<td></td>
<td>HHVM 3.2.0</td>
<td>5.353 ± 0.262</td>
</tr>
<tr>
<td></td>
<td>ZEND 5.4.4</td>
<td>10.406 ± 0.105</td>
</tr>
<tr>
<td>Composed</td>
<td>Bridge</td>
<td>0.936 ± 0.038</td>
</tr>
</tbody>
</table>
Datatype conversion

Diagram:

```
+---+       +---+   +---+
| PHPRoot | -> | PHPObject | PHPInt | PHPFunc |
+---+       +---+   +---+
```
Datatype conversion

```
<table>
<thead>
<tr>
<th>PHPRoot</th>
</tr>
</thead>
<tbody>
<tr>
<td>PHPObject</td>
</tr>
<tr>
<td>PHPInt</td>
</tr>
<tr>
<td>PHPFunc</td>
</tr>
<tr>
<td>PyRoot</td>
</tr>
<tr>
<td>PyObject</td>
</tr>
<tr>
<td>PyInt</td>
</tr>
<tr>
<td>PyFunc</td>
</tr>
</tbody>
</table>
```

HTTP://SOFT-DEV.ORg/
Datatype conversion: primitive types

PHP

Python
Datatype conversion: primitive types

<table>
<thead>
<tr>
<th>PHP</th>
<th>Python</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 : PHPInt</td>
<td></td>
</tr>
</tbody>
</table>
Datatype conversion: primitive types

PHP

2 : PHPInt

Python

2 : PyInt

17/22 HTTP://SOFT-DEV.ORG/
Datatype conversion: user types

PHP

Python
Datatype conversion: user types

- **PHP**
  - `o : PHPObject`

- **Python**
Datatype conversion: user types

Diagram:

```
PyRoot

PyObject  PyInt  PyFunc
```

HTTP://SOFT-DEV.ORg/
Datatype conversion: user types

```
PyRoot

PyObject  PyInt  PyFunc

ProxiedPHPOObject
```
Datatype conversion: user types

PyRoot

PyObject

PyInt

PyFunc

ProxiedPHPObject

php_obj : PHPObject
Datatype conversion: user types

PHP

```
$object = \PHP\Object;
```

Python

```
$object = \Python\Object;
```
Datatype conversion: user types

PHP

\texttt{o : \texttt{PHPObject}}

Python

\texttt{:ProxiedPHPObject}
Datatype conversion: user types

PHP

```
o : PHPObj ect
```

Python

```
: ProxiedPHPObj ect
```

\[
\text{php\_obj}
\]
Datatype conversion: user types

PHP

\( o : \text{PHPObject} \)

Python

\( \text{ProxiedPHPObject} \)

\( \text{php_obj} \)

\text{Immutable field}
Some thoughts

- Critical: single meta-language (e.g. RPython / Truffle).
Some thoughts

- Critical: single meta-language (e.g. RPython / Truffle).
- Simplicity: good performance, yet understandable.
Some thoughts

- Critical: single meta-language (e.g. RPython / Truffle).
- Simplicity: good performance, yet understandable.
- Immutable wrappers give near-native performance.
Some thoughts

- Critical: single meta-language (e.g. RPython / Truffle).
- Simplicity: good performance, yet understandable.
- Immutable wrappers give near-native performance.
- **Whole new world of challenges for language designers & formalisers.**
What can we use this for?
What can we use this for?

First-class languages
What can we use this for?

First-class languages

Language migration
Summary

Bridge

Python
syntax
runtime

PHP
syntax
runtime

Bridge
syntax
runtime
Summary

Language boxes

Python
- Syntax
- Runtime

PHP
- Syntax
- Runtime

Bridge
- Syntax
- Runtime

21/22 HTTP://SOFT-DEV.ORG/
Summary

Language boxes

Composed meta-tracing VMs

Syntax
- Python
  - Syntax
  - Runtime

Syntax
- PHP
  - Syntax
  - Runtime

Bridge
  - Syntax
  - Runtime
Thanks for listening

http://soft-dev.org/