Writing a Custom RPython-based DSL Interpreter to Solve PDEs: Experience and Performance

Sean Fisk, Grand Valley State University

Advisor: Davide Del Vento, NCAR

Huh?

- **DSL:** domain-specific language
  - Often a programming language
- **PDE:** partial differential equation
  - Relates rate of change to continuous variables
  - Can be solved analytically or numerically
- **RPython**
  - A specialized version of Python used for writing interpreters

Goal

Design and build a simple programming language to implement finite difference using stencils.

Finite Difference

- A way to solve partial differential equations
- Frequently used in weather simulations
- Specify a grid, a transformation matrix (stencil), and time step

Stencils – A Simple Example

Matrix to transform (M)

$$\begin{matrix} M_{0,0} & M_{0,1} & M_{0,2} & M_{0,3} & M_{0,4} \\ M_{1,0} & M_{1,1} & M_{1,2} & M_{1,3} & M_{1,4} \\ M_{2,0} & M_{2,1} & M_{2,2} & M_{2,3} & M_{2,4} \\ M_{3,0} & M_{3,1} & M_{3,2} & M_{3,3} & M_{3,4} \\ M_{4,0} & M_{4,1} & M_{4,2} & M_{4,3} & M_{4,4} \end{matrix}$$

Stencil (S)

$$\begin{matrix} S_a & S_b & S_c \\ S_d & S_e & S_f \\ S_g & S_h & S_i \end{matrix}$$

Transformed Matrix

$$M'_{1,1} = M_{1,1} + M_{0,0} \times S_a + \ldots + M_{1,1} \times S_e + \ldots + M_{2,2} \times S_i$$

Other cells use a wrap-around.

Stencils – End Result

Matrix to transform

$$\begin{matrix} 6 & -9 & 10 \\ -9 & -9 & 4 \\ 9 & 5 & -6 \\ 3 & 0 & 9 \end{matrix}$$

Stencil

$$\begin{matrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{matrix}$$

Transformed matrix

$$\begin{matrix} 17 & -13 & 25 \\ -17 & -17 & 9 \\ 24 & 16 & -6 \\ 33 & 27 & 45 \end{matrix}$$

Stencil Language

Instruction	Action
STO R_x V	Store a value in register R_x
ADD R_x V	Add value V to register R_x
PR R_x	Print value in register R_x
CMX M_x A B	Create matrix M_x with dimension (A, B)
SMX M_x ...	Set values of M_x with A × B arguments
SMXF M_x “filename”	Set values of M_x from file “filename”
PMX M_x	Print matrix M_x
PDE M_x M_y	Apply stencil M_x to M_y and store result in M_y
BNE R_x V L	Branch to relative location L if $R_x \neq V$

- Registers use integers
- Matrices use real numbers (floats)
- Matrices must have odd dimensions

Sample Program

```
# Create matrices
CMX 0 3 3
SMXF 0 "sample-programs/matrices/stencil"
CMX 1 3 3
SMXF 1 "sample-programs/matrices/psi"

# Run main PDE loop
STO 0 0
ADD 0 1
PDE 0 1
BNE 0 12 -2
PMX 1
```

PyPy

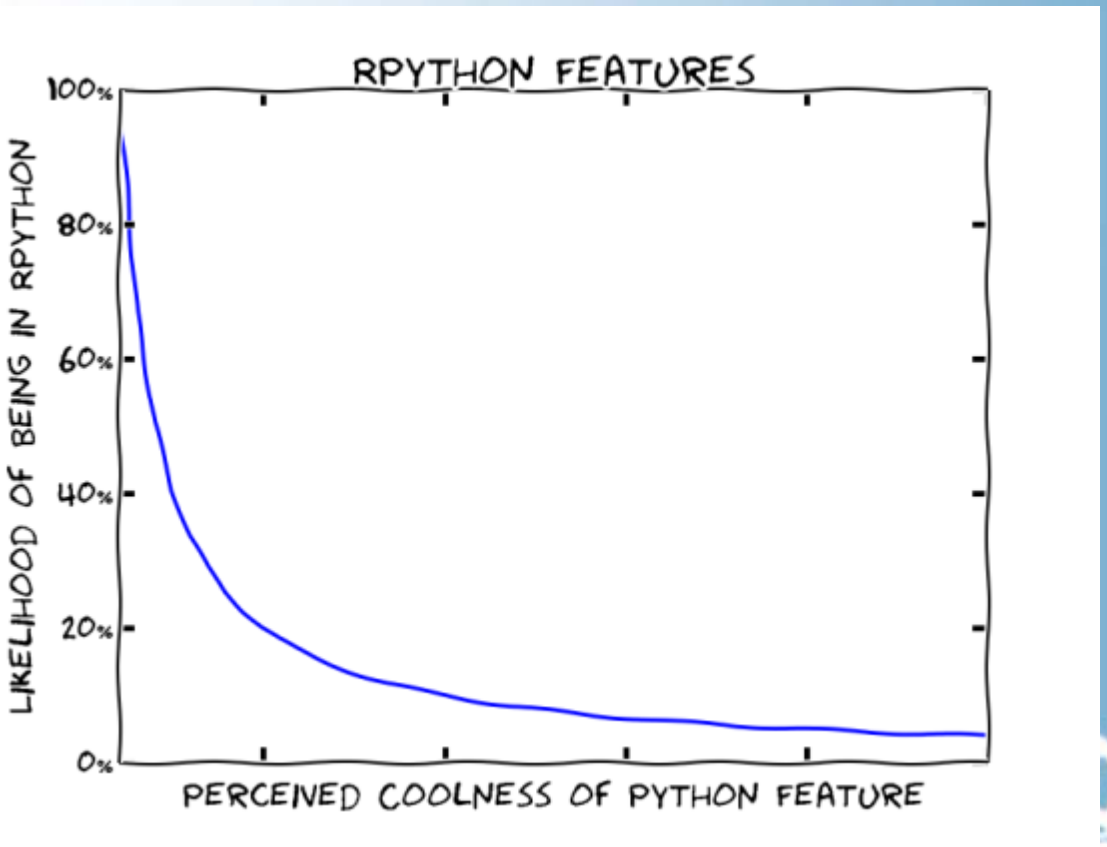
- "Python in Python"
- Continuation of Psyco, a just-in-time compiler for Python
- Umbrella project
  - **RPython:** a toolchain for writing interpreted languages
  - **PyPy:** an implementation of Python in RPython
  - **Just-in-time compiler:** runtime optimization

RPython - Specification

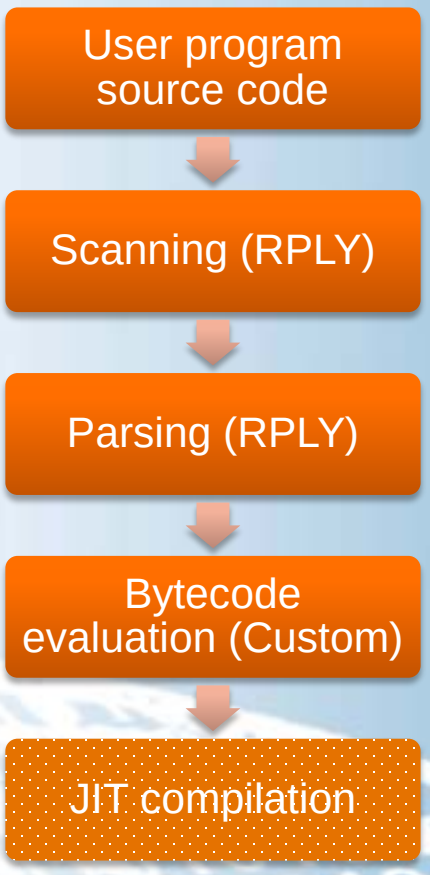
- **RPython is not Python!**
- Restricted subset of Python
- Statically typed, but types are not specified
- `UnionError`
- Use of inheritance

RPython – What is it?

- Whatever the PyPy developers decide to implement.



RPython – Writing an Interpreter



Stages of an Interpreter

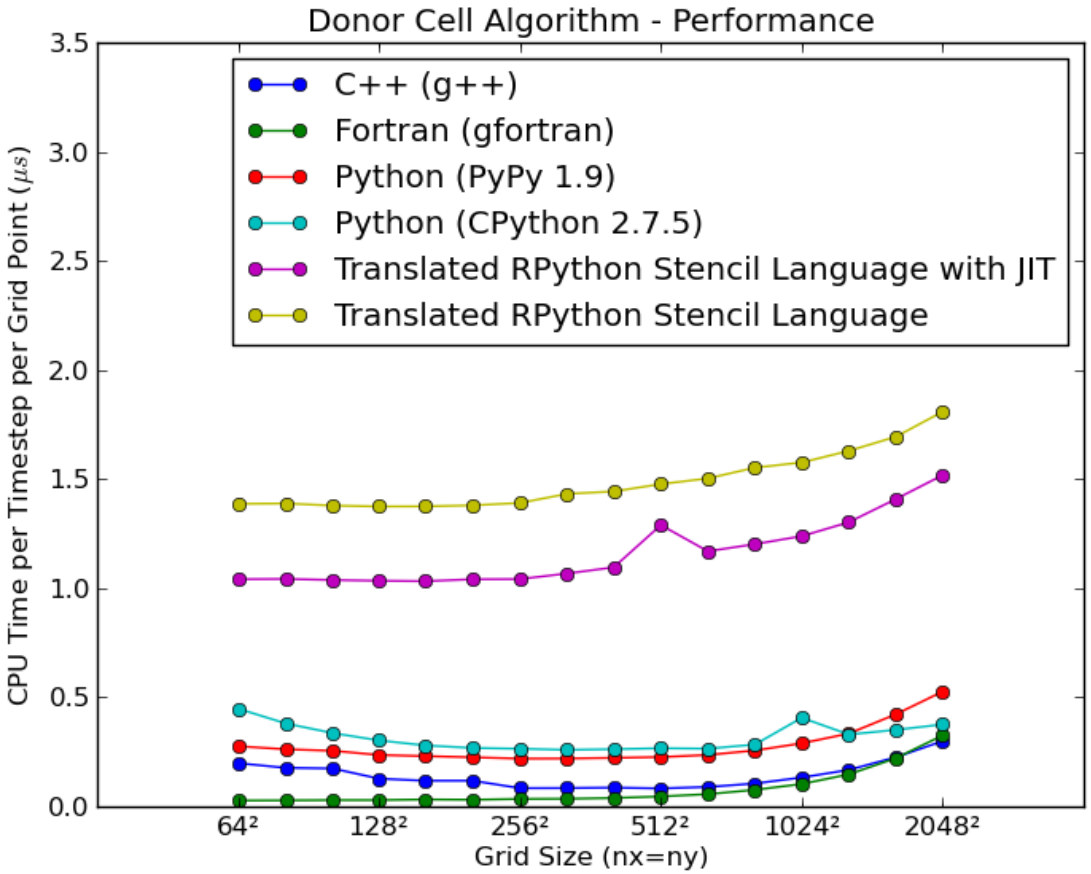
RPython – JIT Compilation

- **JIT**: just-in-time compilation
- Dynamically re-compiles bytecode to machine code at runtime
- Can be faster than statically compiled code
- RPython generates JIT compilers

Donor Cell Algorithm

- A transport algorithm
- Can be implemented as repetitive application of a stencil
- A simplification of the MPDATA algorithm (no correction)

Performance – Comparison



Performance – Why is it slower?

- **Slower than C++/Fortran**
  - Has to interpret a language
- **Slower than Python**
  - Python is more optimized
  - Python uses NumPy (interfaces with C/Fortran)

Software Development

- Almost all code was developed using strict Test-Driven Development
- Stencil language has 142 individually-written tests
- Donor cell implementation runs 632 test cases against existing implementation
- Tools
  - pytest
  - mock
  - tox
  - flake8 (pep8 + pyflakes)

References

- Arabasa, Sylwester et al. (May 13, 2013). “Object-oriented implementations of the MPDATA advection equation solver in C++, Python and Fortran”. In: *arXiv.org*. URL: http://arxiv.org/pdf/1301.1334v2.pdf (visited on 06/27/2013).
- Brown, Andrew (May 6, 2011). *Tutorial Part 2: Adding a JIT*. url: http://morepypy.blogspot.com/2011/04/tutorial- part-2-adding-jit.html (visited on 06/27/2013).
- Brown, Andrew (May 5, 2011b). *Tutorial: Writing an Interpreter with PyPy, Part 1.* url: http://morepypy.blogspot.com/2011/04/tutorial-writing-interpreter-with-pypy.html (visited on 06/27/2013).
- Gaynor, Alex (2013). Topaz 0.1 documentation. Version 0.1. url: http://docs.topazruby.com/en/latest/ (visited on 06/27/2013).
- PyPy Developers (2013). PyPy 2.0.2 documentation. Version 2.0.2. url: http://doc.pypy.org/en/latest/ (visited on 06/27/2013).

Future Ideas

- **Further JIT compiler generation**
- **Parallelization**
  - MPI
  - CUDA
- **Other algorithms**
  - Shallow water

Contact

- **Find the project online**
  - `https://github.com/seanfisk/rpython-stencil-language`
- **Contact me**
  - `<sean@seanfisk.com>`