

# PyPy status talk

Maciej Fijalkowski (merlinux GmbH)  
Dorota Jarecka (University of Warsaw)

EuroSciPy 2009

July 25, 2009

# Contents

- Introduction to PyPy
- Introduction to JIT
- A bit about cloud modelling & EULAG
- JIT & numpy integration
- Future

# PyPy - what's that?



- A Python interpreter written in Python
- A flexible compiler compiling restricted version of Python to lower level platform
- An open source project (MIT license)

# Motivation behind the project

- CPython is not flexible enough
- Python is so much better to write in than C
- Let compiler do the hard work
- Psyco/stackless hard to maintain/extend
- “Psyco consumes one brain per inch of progress”

# Motivation behind the project

- CPython is not flexible enough
- Python is so much better to write in than C
- Let compiler do the hard work
- Psyco/stackless hard to maintain/extend
- “Psyco consumes one brain per inch of progress”

# Motivation - user perspective

- No one should be ever forced to write in C for performance
- ... with maybe some exceptions

# Motivation - user perspective

- No one should be ever forced to write in C for performance
- ... with maybe some exceptions

# Status of PyPy today

- Very compliant python interpreter (passes CPython test suite)
- Can run complex python applications (Django, twisted, ctypes)
- Pretty good garbage collector
- JIT - in progress (more later)

# Python's performance

- Python as a language (not CPython)
- Highly dynamic
- Impossible to efficiently optimize statically
- Dynamic compilation to the rescue

# JIT - motivation

- Psyco hard to maintain and extend
- Python, contrary to popular belief, is a very complex language
- We want to generate JIT out of interpreter's description
- ... instead of writing it by hand

# JIT - motivation

- Psyco hard to maintain and extend
- Python, contrary to popular belief, is a very complex language
- We want to generate JIT out of interpreter's description
- ... instead of writing it by hand

# JIT details

- Interpreter + JIT automatically generated from the interpreter
- Tracing JIT (a-la tracemonkey)
- Written in a high-level language

# JIT - status

- Simple numeric stuff - 20-30x performance boost
- ... which is similar to unoptimized C (gcc -O0)
- only x86 backend so far (ie no x87)
- More work needed
- Only about 2 man-year put so far

# JIT - status

- Simple numeric stuff - 20-30x performance boost
- ... which is similar to unoptimized C (gcc -O0)
- only x86 backend so far (ie no x87)
- More work needed
- Only about 2 man-year put so far

# JIT - status

- Simple numeric stuff - 20-30x performance boost
- ... which is similar to unoptimized C (gcc -O0)
- only x86 backend so far (ie no x87)
- More work needed
- Only about 2 man-year put so far

# Cloud modelling

Cloud processes cover tremendous range of scales, from thousands of kilometers to a fraction of a cm...

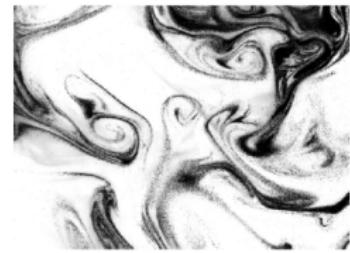
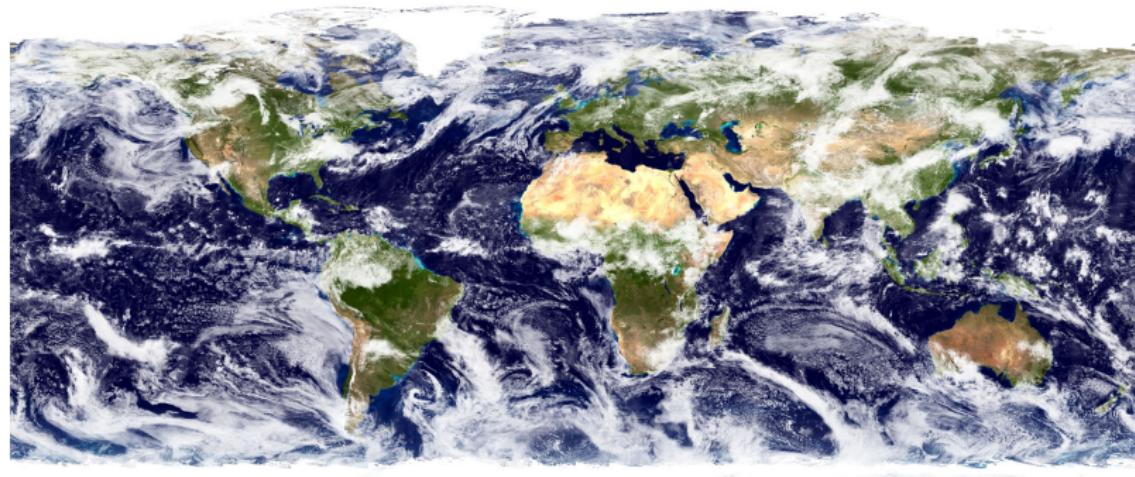


Figure 9. An example image from the cloud chamber. Dark pixels are occupied by cloud droplets. Image covers a physical area of  $\sim 9 \times 6 \text{ cm}^2$ .

# Global climate model

Horizontal grid size of Global Climate Models is  $\sim 100\text{km}$



Clouds have to be parametrized

# Large Eddy Simulation (LES) Models

- processes, which scales are range of 100m could be calculated explicitly
- processes of smaller spatial scale, (e.g. turbulent mixing) have to be still parametrized



Malinowski et al. 2008

# EULAG

- developed in National Center for Atmospheric Science (NCAR), Boulder, Colorado, since '80
- written in Fortran 77
- many different developers and users
- lack of version control, good documentation and bug tracking system

# EULAG - data size

- model time step - 3s, total time of simulations - 6h
- horizontal/vertical grid size 25m/25m number of horizontal/vertical points - 128/121

# Analyzing model data

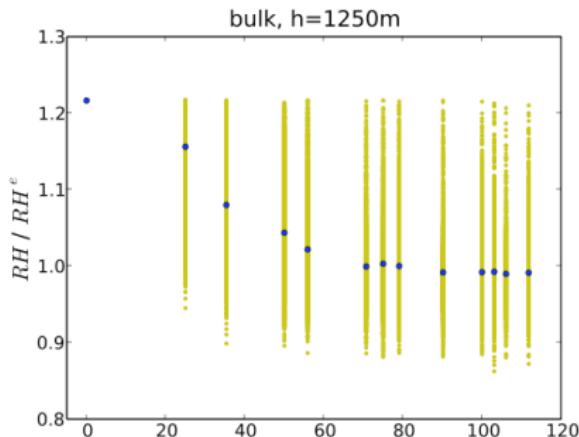


Figure: Distance to the cloud edge

- checkin amount of liquid water in each box
- checkin for each “dry” box, distance to the cloud edge

# Data used for analysis

- written every 4 minutes
- last 4 hours of the simulation
- around 8 basic variables, e.g. velocities, temperature
- For each variable:  $60 * 121 * 128 * 128$  values

# Data postprocessing

- Short programs (~200 python LOC)
- Relatively simple logic
- Numpy for operations, matplotlib for output
- Massive datasets
- Sometimes require walking array elements
- Sometimes slow .....

# Data postprocessing - solution

- Implement minimal version of numeric for PyPy
- Use the JIT for speedups

# Results

- ~10x speedups over pure-python version on top of CPython
- Still about 4x slower than C or matrix operations
- ... but not everything can be expressed using matrix operations
- Due to limitations, programs needs to be carefully crafted

# Results

- ~10x speedups over pure-python version on top of CPython
- Still about 4x slower than C or matrix operations
- ... but not everything can be expressed using matrix operations
- Due to limitations, programs needs to be carefully crafted

# JIT & Numpy

- Just started really
- 138 LOC
- only ints and simple operations

# JIT & Numpy status

- No support from JIT **at all**
- General JIT progress gives speedups with numpy
- More work on numpy needed, but it's all very simple

# JIT & Numpy - future

- Possible to optimize beyond C implementation
- Matrix operations like:  $a + b + c$  should not allocate immediate results
- Use of SSE operations can yield even bigger speedups (or whatever architecture permits)
- Make sure that unmodified programs run faster
- Future looking pretty good

# Future directions

- Work more on JIT in general (inlining, x86\_64 backend, etc.)
- Produce a release by the end of the year
- Release should speed up real-world program
- Work on numpy as time permits
- Seek more funding

# How you can help?

- PyPy is open source after all
- Very friendly to newcomers
- A lot of promising fields to work on
- We also seek funding or research grants

# Thank you! Questions?

- **PyPy:**

<http://codespeak.net/pypy>

<http://morepypy.blogspot.com>

pypy-dev@codespeak.net

- **Cloud modelling:**

dorota@igf.fuw.edu.pl