

# PyPy and the future of the Python ecosystem

Romain Guillebert



Fosdem 2015

January 31st, 2015

# Intro

- ▶ @rguillebert
- ▶ PyPy contributor for 4 years
- ▶ Library compatibility is one of my main interests
  - ▶ Cython backend for PyPy
  - ▶ NumPyPy
  - ▶ PyMetabiosis
- ▶ Hire me
- ▶ How can we get better implementations ?
- ▶ Without throwing away our language features and libraries

# Current situation (1/3)

- ▶ CPython is by far the most popular implementations
  - ▶ *Poor performance*
  - ▶ *No way to use multiple cores in a single process*
- ▶ PyPy has a fairly small marketshare
  - ▶ *Better performance*
  - ▶ *PyPy-STM is a work in progress*
- ▶ According to PyPI stats, other implementations are virtually unused

## Current situation (2/3)

- ▶ Go is pretty fast and is great at concurrency
- ▶ Javascript is pretty fast
- ▶ Even PHP is fast these days...

## Current situation (3/3)

- ▶ It's pretty hard to switch between implementations because of C extensions ()
- ▶ C extensions are very useful but CPython can't evolve because of them
- ▶ PyPy can evolve but has partial support of C extensions
- ▶ CPython keeps its users captive with C extensions
- ▶ More competition between implementations would benefit us

## Current situation (3/3)

- ▶ It's pretty hard to switch between implementations because of C extensions ()
- ▶ C extensions are very useful but CPython can't evolve because of them
- ▶ PyPy can evolve but has partial support of C extensions
- ▶ CPython keeps its users captive with C extensions
- ▶ More competition between implementations would benefit us

# Why can't other implementations implement the C API

- ▶ Libraries use more than the official API (Cython)
- ▶ The official API makes assumptions on how the virtual machine is written
  - ▶ *For example, the C API assumes that the virtual machine uses naive reference counting as its garbage collector*
  - ▶ *Naive reference counting is known for being inefficient and makes removing the GIL really hard (Python 1.4)*
- ▶ The C API itself is against performance and concurrency

# C APIs in other languages

- ▶ JNI / V8
- ▶ Lua / Julia



# Can we implement a similar API ?

- ▶ Yes !
- ▶ Not that many changes to the C API are required
- ▶ It's even possible to have a C API written in pure Python with CFFI
- ▶ Designing it to make everyone happy is harder than to actually implement it
- ▶ Making people port their extensions is hard
- ▶ CPython would need to keep both APIs, at least for a while

# Where does PyPy fit in this ?

- ▶ The most flexible implementation
- ▶ RPython

# The Jit

- ▶ [speed.pypy.org](http://speed.pypy.org)
- ▶ 6.9 times faster than CPython on our benchmarks
- ▶ Competes with other fast dynamic languages
- ▶ Pay the cost of what you use

# CFFI

- ▶ Interacting with C code is very important to the Python community
- ▶ CFFI allows you to call C code and expose Python functions to C
- ▶ Very fast on PyPy
- ▶ As powerful as the C API

# STM

- ▶ Removing the GIL
- ▶ Without having to deal with threads and locks
- ▶ Still allows you to share memory between threads

# Short term C extension support

- ▶ We can bridge PyPy and CPython and let CPython deal with C extensions
- ▶ PyMetabiosis demo
- ▶ This will allow us to bring the entire scientific stack in a very short amount of time

Thank you

Questions ?