PyPy 2016

Matti Picus

Scipy Israel 2016

May 2nd, 2016

PyPy 2016

Python means four things:

Python is

- A syntax or six *(2 * 3 = 6)*
- An interpreter to run code written in the syntax
- A set of standard libraries shipped with the interpreter
- A vibrant number of communities that share code

Python means four things:

Python is

- A syntax or six *(2 * 3 = 6)*
- An interpreter to run code written in the syntax
- A set of standard libraries shipped with the interpreter
- A vibrant number of communities that share code

Python means four things:

Python is

- A syntax or six *(2 * 3 = 6)*
- An interpreter to run code written in the syntax
- A set of standard libraries shipped with the interpreter
- A vibrant number of communities that share code

Python means four things:

Python is

- A syntax or six *(2 * 3 = 6)*
- An interpreter to run code written in the syntax
- A set of standard libraries shipped with the interpreter
- A vibrant number of communities that share code

Techniques to achieve performant Python

- Write better code
    - string concatenation
    - attribute lookup
- Rewrite your code in C
- Rewrite your code in Cython
- Add accelators like Numba
- Use PyPy

Techniques to achieve performant Python

- Write better code
    - string concatenation
    - attribute lookup
- Rewrite your code in C
- Rewrite your code in Cython
- Add accelators like Numba
- Use PyPy

Techniques to achieve performant Python

- Write better code
    - string concatenation
    - attribute lookup
- Rewrite your code in C
- Rewrite your code in Cython
- Add accelators like Numba
- Use PyPy

Techniques to achieve performant Python

- Write better code
    - string concatenation
    - attribute lookup
- Rewrite your code in C
- Rewrite your code in Cython
- Add accelators like Numba
- Use PyPy

Techniques to achieve performant Python

- Write better code
  - string concatenation
  - attribute lookup
- Rewrite your code in C
- Rewrite your code in Cython
- Add accelators like Numba
- Use PyPy

Techniques to achieve performant Python

- Write better code
  - string concatenation
  - attribute lookup
- Rewrite your code in C
- Rewrite your code in Cython
- Add accelators like Numba
- Use PyPy

Techniques to achieve performant Python

- Write better code
  - string concatenation
  - attribute lookup
- Rewrite your code in C
- Rewrite your code in Cython
- Add accelators like Numba
- Use PyPy

PyPy

- PyPy is an interpreter written in RPython
- It ships with the standard library
- Speed is one of its main advantages
- Compatible (mostly) via pip install
- Not the only alternative interpreter

PyPy

- PyPy is an interpreter written in RPython
- It ships with the standard library
- Speed is one of its main advantages
- Compatible (mostly) via pip install
- Not the only alternative interpreter

PyPy

- PyPy is an interpreter written in RPython
- It ships with the standard library
- Speed is one of its main advantages
- Compatible (mostly) via pip install
- Not the only alternative interpreter

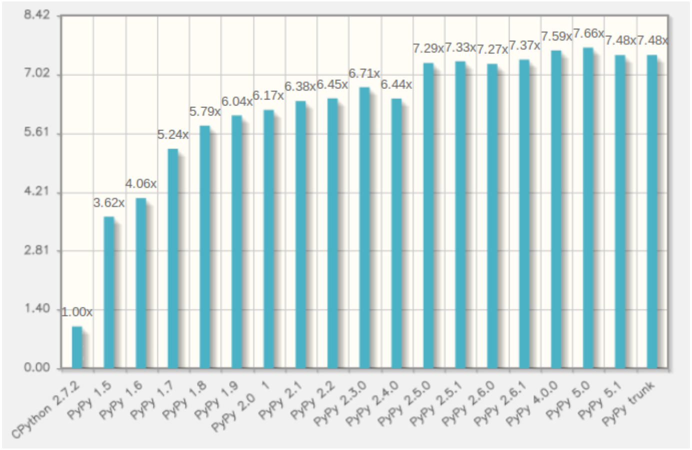
PyPy

- PyPy is an interpreter written in RPython
- It ships with the standard library
- Speed is one of its main advantages
- Compatible (mostly) via pip install
- Not the only alternative interpreter

PyPy

- PyPy is an interpreter written in RPython
- It ships with the standard library
- Speed is one of its main advantages
- Compatible (mostly) via pip install
- Not the only alternative interpreter

Speed (Applause)



Speed continued

- Benchmarking, statistics, politics
- Did I mention warmup time?

Speed continued

- Benchmarking, statistics, politics
- Did I mention warmup time?

How ?

- Tracing Just-In-Time compiler
- Optimizes loops
- Traces one iteration of a loop
- Produces a linear trace of execution
- The trace is then **optimized** and compiled

How ?

- Tracing Just-In-Time compiler
- Optimizes loops
- Traces one iteration of a loop
- Produces a linear trace of execution
- The trace is then **optimized** and compiled

How ?

- Tracing Just-In-Time compiler
- Optimizes loops
- Traces one iteration of a loop
- Produces a linear trace of execution
- The trace is then **optimized** and compiled

How ?

- Tracing Just-In-Time compiler
- Optimizes loops
- Traces one iteration of a loop
- Produces a linear trace of execution
- The trace is then **optimized** and compiled

How ?

- Tracing Just-In-Time compiler
- Optimizes loops
- Traces one iteration of a loop
- Produces a linear trace of execution
- The trace is then **optimized** and compiled

Why is this fast?

- Inlining
- Promotion
- Unrolling
- Strategies
  - Convert sequences to arrays
  - Vectorization

Why is this fast?

- Inlining
- Promotion
- Unrolling
- Strategies
  - Convert sequences to arrays
  - Vectorization

Why is this fast?

- Inlining
- Promotion
- Unrolling
- Strategies
    - Convert sequences to arrays
    - Vectorization

Why is this fast?

- Inlining
- Promotion
- Unrolling
- Strategies
    - Convert sequences to arrays
    - Vectorization

Why is this fast?

- Inlining
- Promotion
- Unrolling
- Strategies
  - Convert sequences to arrays
  - Vectorization

Why is this fast?

- Inlining
- Promotion
- Unrolling
- Strategies
  - Convert sequences to arrays
  - Vectorization

Prove It

- profiling
- jitviewer

Prove It

- profiling
- jitviewer

Why not PyPy?

- Python and the community
- Third-party library support
- No easy packaging (like Winpython or Anaconda)
  - Opportunity???

Why not PyPy?

- Python and the community
- Third-party library support
- No easy packaging (like Winpython or Anaconda)
    - Opportunity???

Why not PyPy?

- Python and the community
- Third-party library support
- No easy packaging (like Winpython or Anaconda)
    - Opportunity???

Why not PyPy?

- Python and the community
- Third-party library support
- No easy packaging (like Winpython or Anaconda)
    - Opportunity???

PyPy and C (1/3)

- PyPy and CFFI (Armin Rigo, Maciej Fijałkowski)
- CFFI is easy, just massage the headers and that's it
- Use CFFI to call python from C
    - This means you can create your own C API in pure Python !

PyPy and C (1/3)

- PyPy and CFFI (Armin Rigo, Maciej Fijałkowski)
- CFFI is easy, just massage the headers and that's it
- Use CFFI to call python from C
  - This means you can create your own C API in pure Python !

PyPy and C (1/3)

- PyPy and CFFI (Armin Rigo, Maciej Fijałkowski)
- CFFI is easy, just massage the headers and that's it
- Use CFFI to call python from C
    - This means you can create your own C API in pure Python !

PyPy and C (1/3)

- PyPy and CFFI (Armin Rigo, Maciej Fijałkowski)
- CFFI is easy, just massage the headers and that's it
- Use CFFI to call python from C
    - This means you can create your own C API in pure Python !

PyPy and C (2/3)

- CFFI enables embedded Python (and PyPy) in a C application (uWSGI)
- Very fast on PyPy, fast enough on CPython

PyPy and C (2/3)

- CFFI enables embedded Python (and PyPy) in a C application (uWSGI)
- Very fast on PyPy, fast enough on CPython

PyPy and C (3/3)

- What about C-API (glad you asked)
- Actively worked on right now

PyPy and C (3/3)

- What about C-API (glad you asked)
- Actively worked on right now

Python C API

- Leaks way too many implementation details (refcounting, PyObject structure fields)
- C allows you to cheat (private, read-only)
- Makes it hard to improve Python while supporting 100% of the API
- PyPy 5.0 introduced a major rewrite
- Hint - good things are coming

Python C API

- Leaks way too many implementation details (refcounting, PyObject structure fields)
- C allows you to cheat (private, read-only)
- Makes it hard to improve Python while supporting 100% of the API
- PyPy 5.0 introduced a major rewrite
- Hint - good things are coming

Python C API

- Leaks way too many implementation details (refcounting, PyObject structure fields)
- C allows you to cheat (private, read-only)
- Makes it hard to improve Python while supporting 100% of the API
- PyPy 5.0 introduced a major rewrite
- Hint - good things are coming

Python C API

- Leaks way too many implementation details (refcounting, PyObject structure fields)
- C allows you to cheat (private, read-only)
- Makes it hard to improve Python while supporting 100% of the API
- PyPy 5.0 introduced a major rewrite
- Hint - good things are coming

Python C API

- Leaks way too many implementation details (refcounting, PyObject structure fields)
- C allows you to cheat (private, read-only)
- Makes it hard to improve Python while supporting 100% of the API
- PyPy 5.0 introduced a major rewrite
- Hint - good things are coming

NumPyPy

- `https://bitbucket.org/pypy/numpy` + pypy
- I have been working on it since 2011, together with many others
- Replaces ndarray, umath with builtin modules
- ~85% of the numpy tests are passing, on all platforms
- Most of numpy is there: object dtypes, ufuncs
- linalg, fft, random all via cffi
- Should be as fast as Numpy, faster for smaller arrays

NumPyPy

- `https://bitbucket.org/pypy/numpy` + pypy
- I have been working on it since 2011, together with many others
- Replaces ndarray, umath with builtin modules
- ~85% of the numpy tests are passing, on all platforms
- Most of numpy is there: object dtypes, ufuncs
- linalg, fft, random all via cffi
- Should be as fast as Numpy, faster for smaller arrays

NumPyPy

- `https://bitbucket.org/pypy/numpy` + pypy
- I have been working on it since 2011, together with many others
- Replaces ndarray, umath with builtin modules
- ~85% of the numpy tests are passing, on all platforms
- Most of numpy is there: object dtypes, ufuncs
- linalg, fft, random all via cffi
- Should be as fast as Numpy, faster for smaller arrays

NumPyPy

- `https://bitbucket.org/pypy/numpy` + pypy
- I have been working on it since 2011, together with many others
- Replaces ndarray, umath with builtin modules
- ~85% of the numpy tests are passing, on all platforms
- Most of numpy is there: object dtypes, ufuncs
- linalg, fft, random all via cffi
- Should be as fast as Numpy, faster for smaller arrays

NumPyPy

- `https://bitbucket.org/pypy/numpy` + pypy
- I have been working on it since 2011, together with many others
- Replaces ndarray, umath with builtin modules
- ~85% of the numpy tests are passing, on all platforms
- Most of numpy is there: object dtypes, ufuncs
- linalg, fft, random all via cffi
- Should be as fast as Numpy, faster for smaller arrays

NumPyPy

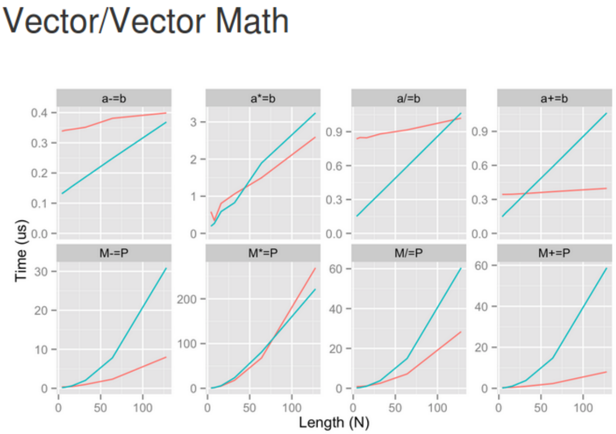
- `https://bitbucket.org/pypy/numpy` + pypy
- I have been working on it since 2011, together with many others
- Replaces ndarray, umath with builtin modules
- ~85% of the numpy tests are passing, on all platforms
- Most of numpy is there: object dtypes, ufuncs
- linalg, fft, random all via cffi
- Should be as fast as Numpy, faster for smaller arrays

NumPyPy

- `https://bitbucket.org/pypy/numpy` + pypy
- I have been working on it since 2011, together with many others
- Replaces ndarray, umath with builtin modules
- ~85% of the numpy tests are passing, on all platforms
- Most of numpy is there: object dtypes, ufuncs
- linalg, fft, random all via cffi
- Should be as fast as Numpy, faster for smaller arrays

NumPyPy performance

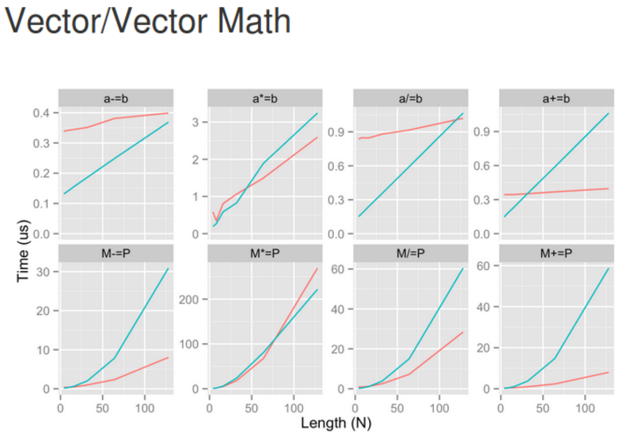
- From `http://rpubs.com/mikefc/60129`



- numpypy in blue, numpy in red

NumPyPy performance

- From `http://rpubs.com/mikefc/60129`



- numpypy in blue, numpy in red

NumPyPy future

- Lazy evaluation ?
- But what about SciPy?

NumPyPy future

- Lazy evaluation ?
- But what about SciPy?

PyMetabiosis

- https://github.com/rguillebert/pymetabiosis
- Proof of concept (Romain Guillebert)
- Allows you to use any CPython module on PyPy (scipy for example)
- Embeds CPython into PyPy with CFFI
- Numpy arrays can be shared between PyPy and CPython

PyMetabiosis

- `https://github.com/rguillebert/pymetabiosis`
- Proof of concept (Romain Guillebert)
- Allows you to use any CPython module on PyPy (scipy for example)
- Embeds CPython into PyPy with CFFI
- Numpy arrays can be shared between PyPy and CPython

PyMetabiosis

- `https://github.com/rguillebert/pymetabiosis`
- Proof of concept (Romain Guillebert)
- Allows you to use any CPython module on PyPy (scipy for example)
- Embeds CPython into PyPy with CFFI
- Numpy arrays can be shared between PyPy and CPython

PyMetabiosis

- `https://github.com/rguillebert/pymetabiosis`
- Proof of concept (Romain Guillebert)
- Allows you to use any CPython module on PyPy (scipy for example)
- Embeds CPython into PyPy with CFFI
- Numpy arrays can be shared between PyPy and CPython

PyMetabiosis

- `https://github.com/rguillebert/pymetabiosis`
- Proof of concept (Romain Guillebert)
- Allows you to use any CPython module on PyPy (scipy for example)
- Embeds CPython into PyPy with CFFI
- Numpy arrays can be shared between PyPy and CPython

PyMetabiosis

```
from pymetabiosis import import_module

cpython_virtualenv_path =
    "/tmp/venv/bin/activate_this.py"

builtin = import_module("__builtin__")

# Activate a virtualenv for the cpython interpreter
builtin.execfile(cpython_virtualenv_path,
    {"__file__" : cpython_virtualenv_path}
)

pylab = import_module("matplotlib.pylab")

pylab.plot([1, 2, 3, 4])
pylab.show()
```

JitPy

- http://jitpy.readthedocs.io
- Proof of concept (Maciej Fijałkowski)
- Embeds PyPy into CPython
- Provides a decorator that allows you to run specific functions on PyPy
- Is used the same way as numba, but different performance characteristics

JitPy

- `http://jitpy.readthedocs.io`
- Proof of concept (Maciej Fijałkowski)
- Embeds PyPy into CPython
- Provides a decorator that allows you to run specific functions on PyPy
- Is used the same way as numba, but different performance characteristics

JitPy

- `http://jitpy.readthedocs.io`
- Proof of concept (Maciej Fijałkowski)
- Embeds PyPy into CPython
- Provides a decorator that allows you to run specific functions on PyPy
- Is used the same way as numba, but different performance characteristics

JitPy

- `http://jitpy.readthedocs.io`
- Proof of concept (Maciej Fijałkowski)
- Embeds PyPy into CPython
- Provides a decorator that allows you to run specific functions on PyPy
- Is used the same way as numba, but different performance characteristics

JitPy

- `http://jitpy.readthedocs.io`
- Proof of concept (Maciej Fijałkowski)
- Embeds PyPy into CPython
- Provides a decorator that allows you to run specific functions on PyPy
- Is used the same way as numba, but different performance characteristics

JitPy

```
import numpy as np
from jitpy import setup
setup('<path-to-pypy-home>')
from jitpy.wrapper import jittify

@jittify(['array', float], float)
def f(a, s):
    r = 0
    for i in xrange(a.shape[0]):
        r += a[i] * s
return s
func(np.arange(10000), 1.2)
```

Future - wouldn’t it be great if

- Improved C extension compatibility
- Native Numpy + Scipy + ...

Future - wouldn't it be great if

- Improved C extension compatibility
- Native Numpy + Scipy + ...

The Future is Now!

- (Applause)
- Native numpy (tweaked) passes 90% of tests
- How to leverage the JIT and NumPyPy?

The Future is Now!

- (Applause)
- Native numpy (tweaked) passes 90% of tests
- How to leverage the JIT and NumPyPy?

The Future is Now!

- (Applause)
- Native numpy (tweaked) passes 90% of tests
- How to leverage the JIT and NumPyPy?

Why this makes sense

- Advantages of RPython
- Advantages of a JIT (vectorization)
- Leveraging this for other dynamic languages

Why this makes sense

- Advantages of RPython
- Advantages of a JIT (vectorization)
- Leveraging this for other dynamic languages

Why this makes sense

- Advantages of RPython
- Advantages of a JIT (vectorization)
- Leveraging this for other dynamic languages

Takeaway

- Get PyPy at pypy.org (or from your favorite distribution)
- Use it in a virtualenv
- Give us feedback (good or bad) #pypy on IRC

Takeaway

- Get PyPy at pypy.org (or from your favorite distribution)
- Use it in a virtualenv
- Give us feedback (good or bad) #pypy on IRC

Takeaway

- Get PyPy at pypy.org (or from your favorite distribution)
- Use it in a virtualenv
- Give us feedback (good or bad) #pypy on IRC

Thank You

Questions ? Examples:

- What about this other interpreter I heard of?
- How can I get involved?
- What about commercial involvement?
- How can I get support?
- What about Python 3.5?

Thank You

Questions ? Examples:

- What about this other interpreter I heard of?
- How can I get involved?
- What about commercial involvement?
- How can I get support?
- What about Python 3.5?

Thank You

Questions ? Examples:

- What about this other interpreter I heard of?
- How can I get involved?
- What about commercial involvement?
- How can I get support?
- What about Python 3.5?

Thank You

Questions ? Examples:

- What about this other interpreter I heard of?
- How can I get involved?
- What about commercial involvement?
- How can I get support?
- What about Python 3.5?

Thank You

Questions ? Examples:

- What about this other interpreter I heard of?
- How can I get involved?
- What about commercial involvement?
- How can I get support?
- What about Python 3.5?