Pythran: Static Compilation of Parallel Scientific Kernels

a.k.a. Python/Numpy compiler for the mass

Proudly made in *Namek* by serge-sans-paille & pbrunet

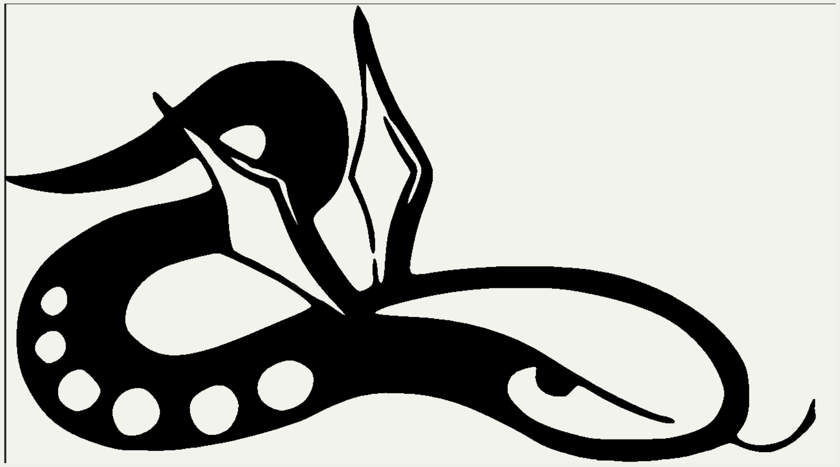
/me

Serge « sans paille » Guelton

```
$ whoami
sguelton
```

- R&D engineer at QuarksLab on compilation for security
- Associate researcher at Télécom Bretagne
- Soldier of fortune for Logilab in OpenDreamKit/HPC

Pythran in a snake shell



- A Numpy-centric Python-to-C++ translator
- A Python code *optimizer*
- A Pythonic C++ library

Core concepts

- Focus on high-level constructs
- Generate ~~clean~~ high level code
- Optimize Python code before generated code
- Vectorization and Paralllelism
- Test, test, test
- Bench, bench, bench

Ask StackOverflow

when you're looking for ***test cases***

http://stackoverflow.com/[...]numba-or-cython-acceleration-in-reaction-diffusion-algorithm

```
import numpy as np
def GrayScott(counts, Du, Dv, F, k):
    n = 300
    U = np.zeros((n+2,n+2), dtype=np.float32)
    V = np.zeros((n+2,n+2), dtype=np.float32)
    u, v = U[1:-1,1:-1], V[1:-1,1:-1]

    r = 20
    u[:] = 1.0
    U[n/2-r:n/2+r,n/2-r:n/2+r] = 0.50
    V[n/2-r:n/2+r,n/2-r:n/2+r] = 0.25
    u += 0.15*np.random.random((n,n))
    v += 0.15*np.random.random((n,n))

    for i in range(counts):
        Lu = (                 U[0:-2,1:-1] +
              U[1:-1,0:-2] - 4*U[1:-1,1:-1] + U[1:-1,2:] +
                               U[2:  ,1:-1] )
```

Thread Summary

OP

My code is slow with Cython and Numba

Best Answer

You need to make all loops explicit

Cython Version

```
cimport cython
import numpy as np
cimport numpy as np

cpdef cythonGrayScott(int counts, double Du, double Dv, double F, double
    cdef int n = 300
    cdef np.ndarray U = np.zeros((n+2,n+2), dtype=np.float_)
    cdef np.ndarray V = np.zeros((n+2,n+2), dtype=np.float_)
    cdef np.ndarray u = U[1:-1,1:-1]
    cdef np.ndarray v = V[1:-1,1:-1]

    cdef int r = 20
    u[:] = 1.0
    U[n/2-r:n/2+r,n/2-r:n/2+r] = 0.50
    V[n/2-r:n/2+r,n/2-r:n/2+r] = 0.25
    u += 0.15*np.random.random((n,n))
    v += 0.15*np.random.random((n,n))
```

Pythran version

Add this line to the original kernel:

```
#pythran export GrayScott(int, float, float, float, float)
```

Timings

```
$ python -m timeit -s 'from grayscott import GrayScott' 'GrayScott(40, 0
10 loops, best of 3: 52.9 msec per loop
$ cython grayscott.pyx
$ gcc grayscott.c `python-config --cflags --libs` -shared -fPIC -o grays
$ python -m timeit -s 'from grayscott import GrayScott' 'GrayScott(40, 0
10 loops, best of 3: 36.4 msec per loop
$ pythran grayscott.py -O3 -march=native
$ python -m timeit -s 'from grayscott import GrayScott' 'GrayScott(40, 0
10 loops, best of 3: 20.3 msec per loop
```

Sample Usage

```
$ pythran input.py # generates input.so
$ pythran input.py -E # generates input.cpp
$ pythran input.py -O3 -fopenmp # parallel!
$ pythran input.py -march=native -Ofast # Esod Mumixam !
```

Benchmarks

https://github.com/serge-sans-paille/numpy-benchmarks

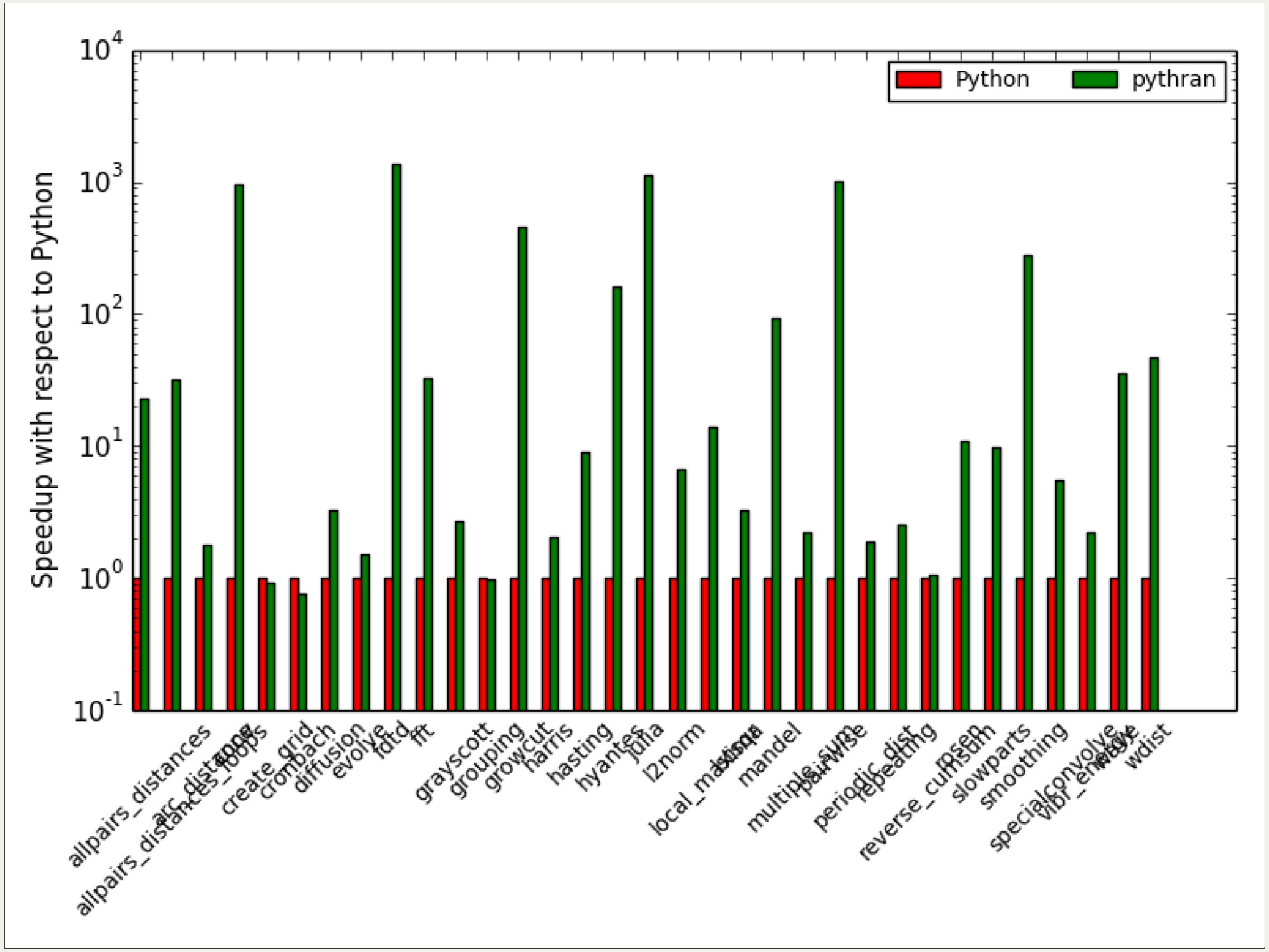
A collection of high-level benchmarks

- Code gathered from StackOverflow + other compiler code base
- Mostly high-level code
- Generate results for CPython, PyPy, Numba, Parakeet, Hope and Pythran

Most kernels are too high level for Numba and Hope…

Benchmarks

no parallelism, no vectorisation (, no fat)



Powered by Strong Engineering

Preprequisite for reproductible science

- 2773 test cases, incl. unit testing, doctest, **CI** (thx Travis!)
- Peer-reviewed code
- Python2.7 and C++11
- IPython Integration
- Linux, OSX (almost okay), Windows (on going)
- User and Developer doc: http://pythonhosted.org/pythran/
- Hosted on https://github.com/serge-sans-paille/pythran
- Releases on PyPi: `$ pip install pythran`
- Custom Debian repo: `$ apt-get install pythran`

OpenDreamKit Challenges

- Better alias analysis & type inference → class support?
- Cython integration
- Sage Notebook integration
- Meet & support OpenDreamKit's community code!

  ⇒ Come and talk with me **today**