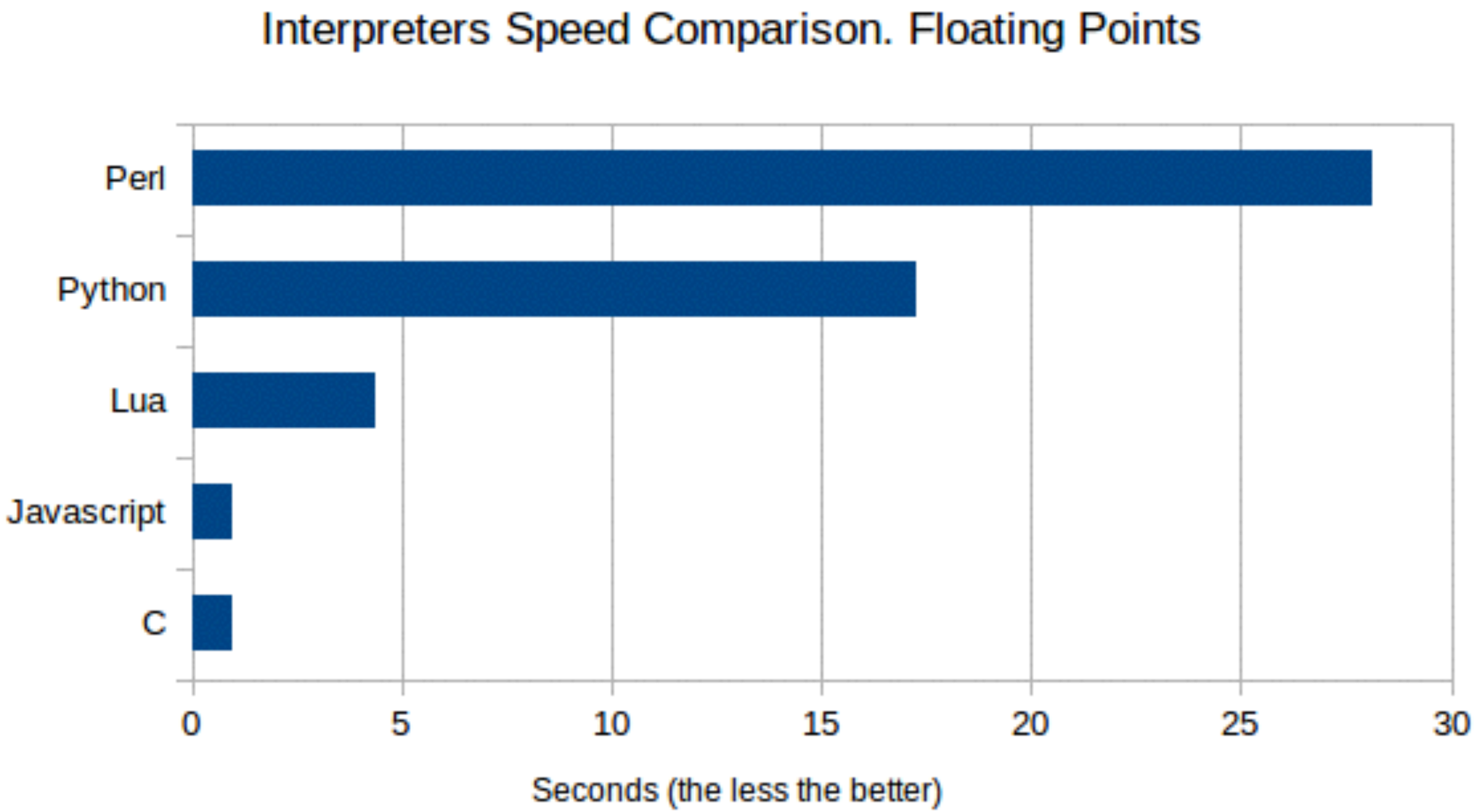
but...


Slow down!
I wanna get there, but I wanna get there alive!

Fortran


C


Interpreters Speed Comparison. Floating Points
Perl
Python
Lua
Javascript
C
0
5
10
15
20
25
30
Seconds (the less the better)

Simplicity

Python

C

C++

Fortran

Performance

Simplicity

Python

Cython

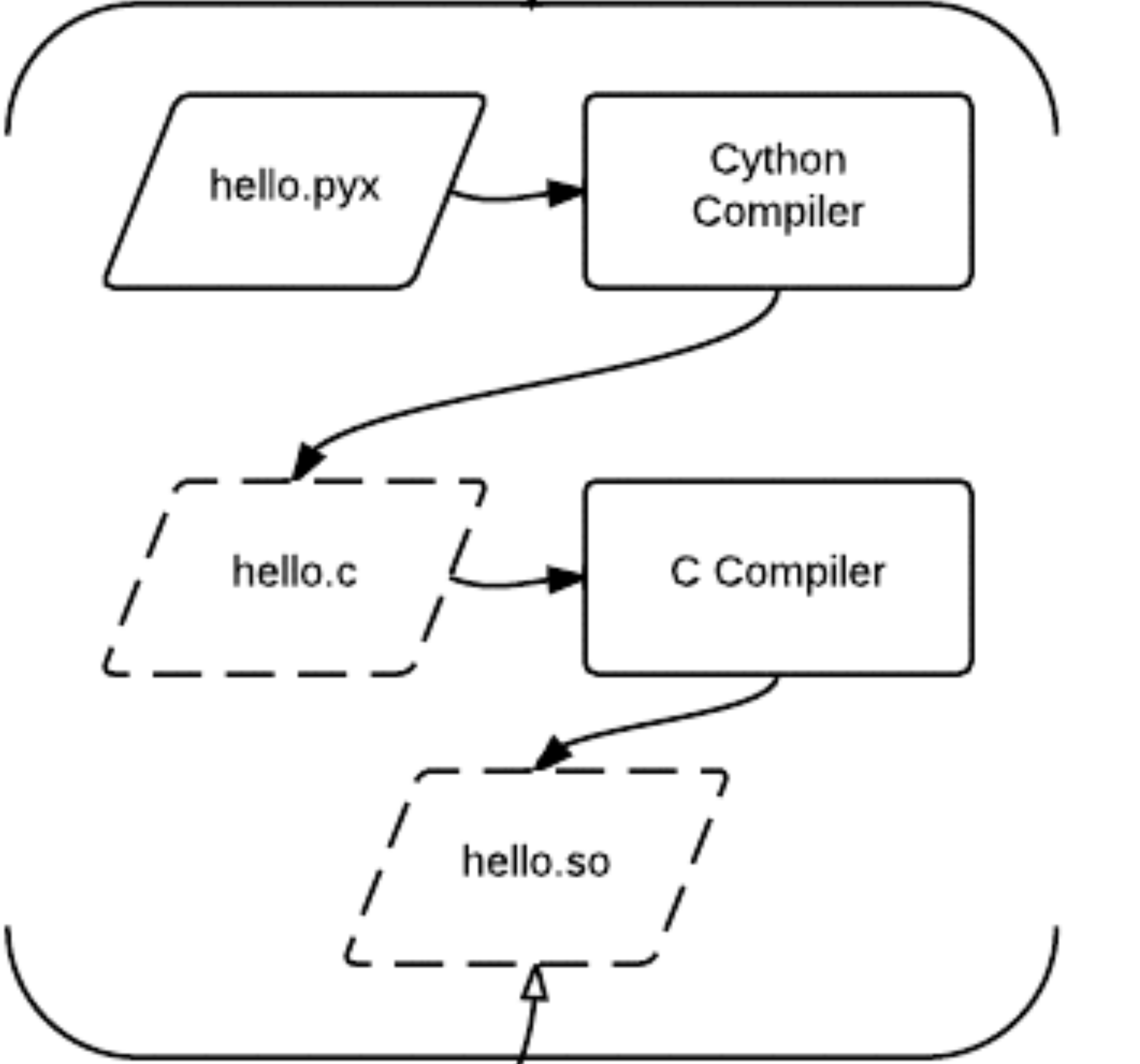
C

C++

Fortran

Performance

Cython


hello.pyx
Cython
Compiler
hello.c
C Compiler
hello.so

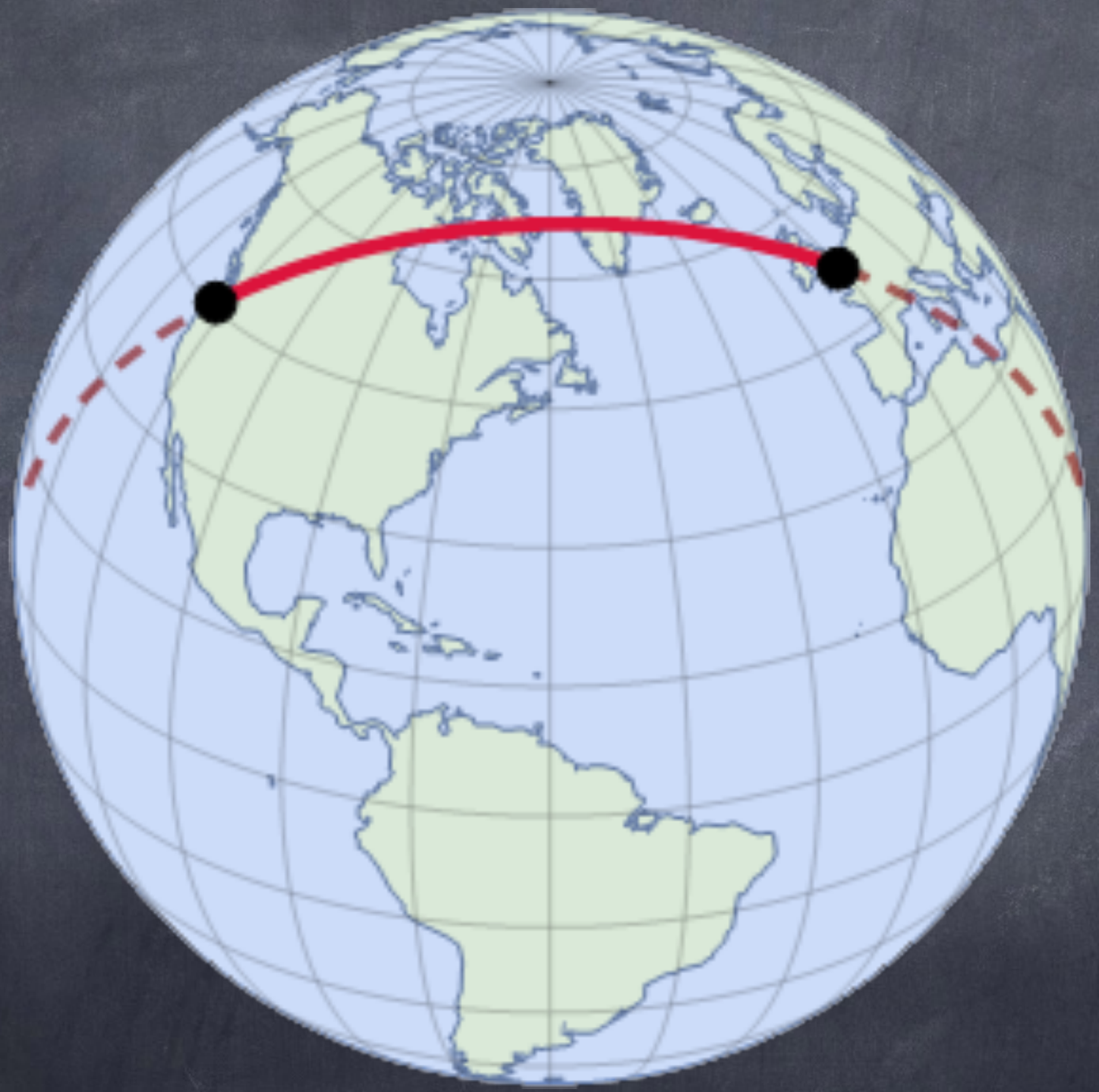
```
# hello.pyx - Python Module,
this code will be translated to
C by Cython.
def say_hello():
    print "Hello World!"
```

```
# launch.py - Python stub loader,
loads the module that was made by
Cython.

# This code is always interpreted,
like normal Python.
# It is not compiled to C.

import hello
hello.say_hello()
```

```c
/* Generated by Cython 0.21 */

#define PY_SSIZE_T_CLEAN
#ifndef CYTHON_USE_PYLONG_INTERNALS
#ifdef PYLONG_BITS_IN_DIGIT
#define CYTHON_USE_PYLONG_INTERNALS 0
#else
#include "pyconfig.h"
#ifdef PYLONG_BITS_IN_DIGIT
#define CYTHON_USE_PYLONG_INTERNALS 1
#else
#define CYTHON_USE_PYLONG_INTERNALS 0
#endif
#endif
#endif
#include "Python.h"
#ifndef Py_PYTHON_H
    #error Python headers needed to compile C extensions, please install development version of Python.
#elif PY_VERSION_HEX < 0x02060000 || (0x03000000 <= PY_VERSION_HEX && PY_VERSION_HEX < 0x03020000)
    #error Cython requires Python 2.6+ or Python 3.2+.
#else
#define CYTHON_ABI "0_21"
#include <stddef.h>
#ifndef offsetof
#define offsetof(type, member) ( (size_t) & ((type*)0) -> member )
#endif
#if !defined(WIN32) && !defined(MS_WINDOWS)
  #ifndef __stdcall
    #define __stdcall
  #endif
  #ifndef __cdecl
    #define __cdecl
  #endif
  #ifndef __fastcall
    #define __fastcall
  #endif
#endif
#ifndef DL_IMPORT
  #define DL_IMPORT(t) t
#endif
#ifndef DL_EXPORT
  #define DL_EXPORT(t) t
#endif
#ifndef PY_LONG_LONG
```

```c
/* "hello.pyx":2
 * # hello.pyx - Python Module, this code will be translated to C by Cython.
 * def say_hello():              # <<<<<<<<<<<<<<
 *     print "Hello World!"
 */

/* Python wrapper */
static PyObject *__pyx_pw_5hello_1say_hello(PyObject *__pyx_self, CYTHON_UNUSED
PyObject *unused); /*proto*/
static PyMethodDef __pyx_mdef_5hello_1say_hello = {"say_hello",
(PyCFunction)__pyx_pw_5hello_1say_hello, METH_NOARGS, 0};
static PyObject *__pyx_pw_5hello_1say_hello(PyObject *__pyx_self, CYTHON_UNUSED
PyObject *unused) {
  PyObject *__pyx_r = 0;
  __Pyx_RefNannyDeclarations
  __Pyx_RefNannySetupContext("say_hello (wrapper)", 0);
  __pyx_r = __pyx_pf_5hello_say_hello(__pyx_self);

  /* function exit code */
  __Pyx_RefNannyFinishContext();
  return __pyx_r;
}
```

Now a real example



To the Notebook

back

Gotchas


PREMATURE OPTIMIZATION
Come on, do it! Do it now! It feels soooo good.

Automatically compiling Cython extension?

Advanced interconnection with C?

Numpy and Cython?

Looking for bottlenecks