interfacing C/C++/FORTRAN

Simone Campagna

pimp your python code/1

Python è un linguaggio relativamente lento, se confrontato con C/C++/FORTRAN. Questo deriva dal fatto che è di alto livello, con tipizzazione dinamica, interpretato.

Ciononostante, può essere utilizzato anche in contesti in cui le prestazioni contano. Il trucco consiste nello scrivere la parte critica del codice in un linguaggio veloce, e poi importarla come modulo python.

pimp your python code/2

In questo modo è possibile sfruttare la potenza di ogni linguaggio nel territorio più naturale:

- la potenza di python nella gestione della complessità (alto livello)
- la potenza di C nella gestione delle prestazioni (basso livello)

python modules/1

È possibile creare moduli python in due maniere:

- scrivendo codice python puro (come abbiamo fatto fino a questo momento)
- scrivendo codice in altri linguaggi (C, C++, FORTRAN, ...) e facendone il *binding* in modo che sia importabile come modulo python.

Dal punto di vista dell'uso di questi moduli, non c'è alcuna differenza!

python modules/2

La maniera nativa per scrivere un modulo python in C consiste nell'usare le python C-API

(http://docs.python.org/c-api/). Si tratta di scrivere codice C attraverso le API di python, che creano la struttura necessaria ad importare gli oggetti e le funzioni come modulo all'interno di python.

È piuttosto complicato!

ctypes/1

Spesso dobbiamo importare in python una libreria (C) già esistente, e non pensata per essere importata in python.

Non è complicato farlo, ci sono vari strumenti.

Ad esempio, il modulo python ctypes consente di chiamare una funzione di una libreria C

ctypes/2

```
>>> from ctypes import *
>>> libc = cdll.LoadLibrary("libc.so.6")
>>> libc.time(None)
1291192748
>>> libc.time(None)
1291192750
>>> libc.time(None)
1291192750
>>> libc.printf("%s %d\n", "Ciao, mondo!", 34)
Ciao, mondo! 34
16
>>>
```

Cython/1

Cython (http://www.cython.org/) è una estensione di python pensata per consentire facilmente di estendere python con moduli scritti in altri linguaggi.

È l'evoluzione di pyrex.

Cython/2

Cython crea automaticamente il binding di oggetti/funzioni C/C++ usando le python C-API. In pratica estende python aggiungendo una sintassi in cui è possibile dichiarare funzioni ed oggetti “C”.

```
cdef extern from "math.h":
    double sin(double)

cdef double f(double x):
    return sin(x*x)
```

Cython/3

Cython ha una sintassi simile a quella di python, con alcune estensioni. Dovrebbe essere un superset di python.

Ciò che produce è codice C/C++ ottimizzato, che può essere compilato per produrre un modulo python.

Cython/4

Vediamo un esempio: viene creato un modulo python mylib.so che è un *wrapping* di una libreria C (mylib) preesistente.

swig/1

Swig (http://www.swig.org/index.php) è un wrapper generator, che consente di creare wrapping di librerie da un linguaggio all'altro; supporta numerosi linguaggi.

swig/2

Con poco lavoro swig consente di ottenere un wrapping python di librerie C o C++, per percentuali alte (senza intervento alcuni elementi della libreria potrebbero essere ignorati).

Per ottenere il wrapping del 100% della libreria, in genere occorre un intervento.

Per ottenere un wrapping con una interfaccia python decente, in genere occorre un intervento.

pyfort/1

Pyfort (http://pyfortran.sourceforge.net/) è uno strumento per costruire interfacce python a funzioni fortran.

pyfort/2

#file *sum.f*

```
function sum(n, x)
  integer n
  real x(n)
  real sum
  ... (the sum code)
end function sum
```

#file *sum.pyf*

```
function sum(n, x)
  ! sum(n, x) -> sum of the real array x(n)
  integer n
  real x(n)
  real sum
end function sum
```

f2py

F2py è uno strumento per eseguire il wrapping di codice fortran all'interno di moduli python. È entrato a far parte di numpy.