Physik auf dem Computer
Cython

Olaf Lenz

Institut für Computerphysik
Universität Stuttgart

Sommersemester 2012

Cython: Python mit C verbinden

- http://cython.org/
- Free, Open Source Software
- Entwickelt seit 2002: *Pyrex* von Greg Ewing
- Fork im Jahre 2007: *Cython*
- Beschleunigung von Python-Modulen
- Aufruf von C-Funktionen aus Python-Programmen

Kompilieren eines Python-Moduls

- Cython erzeugt aus einer Python-Datei eine C-Datei
- Kompilation der C-Datei erzeugt ein *binäres* Python-Modul

```
cython pyfib.py
gcc -shared -c pyfib.c -o pyfib.so -I/usr/include/python2.7
```

- Cython kann *die meisten* Python-Konstrukte übersetzen
- führt zu mittlerer Beschleunigung des Python-Moduls

Kompilieren eines Python-Moduls via distutils

- Einfachere Kompilation über Pythons *Distutils*

```
from distutils.core import setup
from distutils.extension import Extension
from Cython.Distutils import build_ext

setup(
    cmdclass = {'build_ext': build_ext},
    ext_modules = [Extension("pyfib", ["pyfib.py"])]
)
```

- Aufruf

```
python setup.py build_ext --inplace
```

Typen annotieren

- Python-Code langsamer vor allem wegen fehlender Typen
- Cython erlaubt es, im Python-Code *Typinformationen* zu annotieren
- .py → .pyx

fib.py:

```
def fib(n):
  if n <= 1: return 1
  else:
    return fib(n-1) + fib(n-2)
```

fib.pyx:

```
cdef int fib(int n):
  if n <= 1: return 1
  else:
    return fib(n-1) + fib(n-2)
```

- Massive Beschleunigung!
- cdef-Funktionen können im Cython-Code sehr viel schneller aufgerufen werden
- *Caveat emptor*: cdef-Funktionen können von Python nicht mehr direkt aufgerufen werden!

Typen annotieren 2

- Mögliche Typen: float, double, int, bool
- Cython kann auch direkt mit numpy.array umgehen
  http://wiki.cython.org/tutorials/numpy
- cython -a pyfib.py erzeugt HTML-Datei pyfib.html
- Darin sind die Zeilen markiert, die viel Python-Overhead haben

Generated by Cython 0.15.1 on Fri Apr 20 09:04:56 2012

Raw output: pyfib.c

```
 1: def fib(n):
 2:     if n <= 1: return 1
 3:     else: return fib(n-1) + fib(n-2)
 4:
 5: def compute():
 6:     return fib(35)
```

- Zur maximalen Beschleunigung diese Zeilen annotieren!

pyximport

- Cython-Modul noch einfacher kompilieren
- In ein Skript, das ein Cython-Modul importieren soll:

```
import pyximport
pyximport.install()

import cyfib
```

- Kompiliert automatisch das Cython-Modul
- Nachteil: Kein Einfluss auf Kompilationsflags

Anbinden von C-Funktionen an Cython/Python

```
cdef extern from "fibfunc.h":
  int fib(int n)

def pyfib(n):
  return fib(n)
```

- Importiert die C-Funktion fib als cdef-Funktion
- Diese kann von Cython aus aufgerufen werden
- ...aber nicht von Python!
- Deswegen: wrapper-Funktion
- Kochrezept: C-Funktion von Python aus zugänglich machen
  1. .pyx: Importiere die C-Funktionen via cdef extern
  2. .pyx: Schreibe Python-Funktion, die die cdef-Funktion aufruft
  3. setup.py: Schreibe distutils-Datei zum Kompilieren
- Zum Anbinden von C an Python ist also immer eine .pyx-Datei notwendig!