2014-05-09-cython.sagews

May 9, 2014

Contents

1 Math 480b Sage Course

1.1 Overview of Sage

1.2 May 9, 2014

Screencast: REMIND ME
Plan

- Questions
- Cython

1.3 The C Level

- Python is like floating around underwater you can easily float around, etc., but you move slowly.
- C is like flying around in the air you go much more quickly, but can also easily crash and burn
- `Cython` makes it so you can fly

1.4 Motivation

- Browse `http://benchmarksgame.alioth.debian.org/u64/benchmark.php?test=alllang=python3lang2=gccdata=u64` and compare speed of languages on number crunching tasks.

1.5 First benchmark: pi-digits

Lets look at and try out the pi digits benchmark. Here Python and C are pretty close!

But note the use of gmpy, i.e., all the hard work in the python implementation happens at the C level.

```
# sage is way faster... (probably a different algorithm!)
%time s = str(N(pi, digits=10000))
```

```
CPU time: 0.01 s, Wall time: 0.01 s
```

1.6 Next benchmark: Mandelbrot

Oh crap, Python sucks

`http://benchmarksgame.alioth.debian.org/u64/performance.php?test=mandelbrot`

Of all languages, C is the fastest. And Python is almost the worst, being about 86 times slower.

I looked into the Python code, and it does the computation in parallel using 64 processes, even on a single core computer. On a 1600x1600 case, just the time to start those processes and do nothing takes more time than the C program takes to do everything

1.7 Next benchmark: binary-trees

This one is also very depressing for Python.

`http://benchmarksgame.alioth.debian.org/u64/performance.php?test=binarytrees`

C is first and Python is nearly last, being over 50 times slower.

1.8 ???

- why?
- what can we do?
- but Python is so nice and fun.
- Extending Python with C/C++: `https://docs.python.org/2/extending/extending.html`
- Cython: basically makes extending Python with C/C++ way, way easier than the official approach

Lets take a very, very simple example in Python and speed it up using Cython.

1.9 Problem: variance of a list of floating point numbers

variance = the mean of the squares of the difference of each value from the mean = $\sum \frac{1}{n}(x_i - \mu)^2$.

```
# straightforward Python implementation
def var0(v):
    m = float(sum(v))/len(v)
    return sum([(x-m)**2 for x in v])/len(v)  # use ** so this is \
       standard Python
```

```
v = [random() for _ in range(10)]
v
```

```
[0.8589381826391068, 0.8681592440262227, 0.6892358810838731, 0.5800153806503944,
0.6653829882189602, 0.18009338714825207, 0.9077079599252226, 0.8509199973502831,
0.9766987594487242, 0.7177646364529632]
```

```
var0(v)
```

```
0.04736519346871963
```

```
v = [random() for _ in range(10000)]
```

```
%timeit var0(v)
```

```
25 loops, best of 3: 8.63 ms per loop
```

```
# Straightforward Cython implementation
# (if the "show auto-generated" doesn't work properly for you -- with \
  yellow -- restart your project server, since I just fixed a bug in \
  this.)
%cython
def var1(v):
    m = float(sum(v))/len(v)
    return sum([(x-m)**2 for x in v])/len(v)
```

```
https://cloud.sagemath.com/blobs/_projects_74af30b7_ad25_4308_a02e_c71fcd84de6e__sage_temp_
compute19dc0_9366_dir_wbVbwx_a_pyx_0.html?uuid=691f0279-4c77-4c59-bbb6-f2e9c9691f1dShowauto-generatedco
```

```
%timeit var1(v)
```

```
125 loops, best of 3: 3.08 ms per loop
```

```
8.63/3.08
```

```
2.80194805194805
```

```
# Declare some types

%cython
def var2(list v):
    cdef double m, x
    m = float(sum(v))/len(v)
    return sum([(x-m)**2 for x in v])/len(v)
```

```
https://cloud.sagemath.com/blobs/stdsage.html?uuid=f56adb5e-24d0-44df-94b5-b03867d712dcShowauto-generat
```

```
%timeit var2(v)
```

```
625 loops, best of 3: 786 s per loop
```

```
8.63/.786
```

```
10.9796437659033
```

```
# Don't call Python's sum function, but do the sum ourselves; also don't \
   use square which is potentially slow.

%cython
def var3(list v):
    cdef double m, x, s, n
    n = len(v)
    s = 0
    for x in v:
        s += x
    m = s/n
    s = 0
    for x in v:
        s += (x-m)*(x-m)
    return s/n
```

```
https://cloud.sagemath.com/blobs/stdsage.html?uuid=55f5ede0-a2bb-4ed2-a5a1-a6c6b67955fbShowauto-generat
```

```
%timeit var3(v)
```

```
625 loops, best of 3: 209 s per loop
```

```
8.63/.209
```

```
41.2918660287081
```

```
# Next, make our own data type for a list of doubles
```

```
%cython
cdef class DoubleList:
    cdef double* v
    cdef int n
    def __init__(self, v):
        self.n = len(v)
        self.v = <double*> sage_malloc(sizeof(double)*self.n)
        cdef int i
        for i in range(self.n):
            self.v[i] = v[i]

    def variance(self):
        cdef double m, x, s
        cdef int i, n
        n = self.n
        s = 0
        for i in range(n):
            s += self.v[i]
        m = s/n
        s = 0
        for i in range(n):
            x = self.v[i]
            s += (x-m)*(x-m)
        return s/n
```

https://cloud.sagemath.com/blobs/_projects_74af30b7_ad25_4308_a02e_c71fcd84de6e__sage_temp_
compute19dc0_9366_dir_rD2Ets_a_pyx_0.html?uuid=66159b17-c255-42be-908c-f6c61681bd01Showauto-generatedco

```
vd = DoubleList(v)
```

```
%timeit vd.variance()
```

```
625 loops, best of 3: 29.8 s per loop
```

```
# WIN
8.63/.0298
```

```
289.597315436242
```

1.10 Why it is called Cython

http://www.mohawkradio.com/images/merch_images/mens/cython_01.jpg