2014-05-09-cython.sagews

May 9, 2014

Contents

1 Math 480b Sage Course **1**
- 1.1 Overview of Sage . . . 1
- 1.2 May 9, 2014 . . . 1
- 1.3 The C Level . . . 1
- 1.4 Motivation . . . 1
- 1.5 First benchmark: pi-digits . . . 1
- 1.6 Next benchmark: Mandelbrot . . . 2
- 1.7 Next benchmark: binary-trees . . . 2
- 1.8 ??? . . . 2
- 1.9 Problem: variance of a list of floating point numbers . . . 2
- 1.10 Why it is called Cython . . . 5

1 Math 480b Sage Course

1.1 Overview of Sage

1.2 May 9, 2014

Screencast: `http://youtu.be/YrO89QIizxI`

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 = $\sum \frac{1}{n}(x_i - \mu)^2$.

```
%python
# 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
```

```
[0.1241999666440795, 0.7688493160210794, 0.27251914257554644, 0.4032333800198763,
0.9485561288153582, 0.696677854333964, 0.7357727102496147, 0.7339442720461534,
0.5379013599960007, 0.7868187591689886]
```

```
var0(v)
```

```
0.06068201681775176
```

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

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

```
125 loops, best of 3: 4.07 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/interrupt.html?uuid=f2309c10-79da-4755-88d1-5a6b61c8be6bShowauto-gener
```

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

```
125 loops, best of 3: 2.44 ms per loop
```

```
4.07/2.44
```

```
1.66803278688525
```

```
a =float(7)
b = a
c = a
c
```

```
7.0
```

```
del b
```

```
# 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=cd2303b7-5453-464c-bf84-e75460155833Showauto-generat

```
%timeit var2(v)
```
625 loops, best of 3: 807 s per loop

```
4.07/.786
```
5.17811704834606

```
# 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/interrupt.html?uuid=b2a2561a-08e0-4134-8f30-1f7706b0c73fShowauto-gener

```
%timeit var3(v)
```
625 loops, best of 3: 194 s per loop

```
4.07/.209
```
19.4736842105263

```
# 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 __del__(self):
```

```
        sage_free(self.v)
    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/interrupt.html?uuid=22a43544-ea7d-4f57-96b8-441789d55ff8Showauto-gener

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

```
vd
```

```
<_projects_74af30b7_ad25_4308_a02e_c71fcd84de6e__sage_temp_compute19dc0_9366_dir_8g25Tr_a_
pyx_0.DoubleList object at 0x8c9f230>
```

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

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

```
# WIN
4.07/.0298
```

```
136.577181208054
```

1.10 Why it is called Cython

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