加速你的Python代码 【已翻译100%】

英文原文：Speeding Up Your Python Code

标签： Python

renwofei423 推荐于 5年前 (共 7 段, 翻译完成于 03-24) 评论 18

分享

参与翻译 (4人) : 缪斯的情人, fey424, gkgy, crab2313

仅中文 | 中英文对照 | 仅

在我看来，python社区分为了三个流派，分别是python 2.x组织，3.x组织和PyPy组织。这个分类基本上可以归根于类库的兼容性和速度。这篇文章将聚焦于一些通用代码的优化技巧以及编译成C后性能的显著提升，当然我也会给出三大主要python流派运行时间。我的目的不是为了证明一个比另一个强，只是为了让你知道如何在不同的环境下使用这些具体例子作比较。

缪斯的
翻译于 5
4人顶

顶 翻译得不错哦！

使用生成器

一个普遍被忽略的内存优化是生成器的使用。生成器让我们创建一个函数一次只返回一条记录，而不是一次返回所有的记录，如果你正在使用python2.x，这就是你为啥使用xrange替代range或者使用ifilter替代filter的原因。一个很好地例子就是创建一个很大的列表并将它们拼合在一起。

```
import timeit
import random

def generate(num):
while num:
yield random.randrange(10)
num -= 1

def create_list(num):
numbers = []
while num:
numbers.append(random.randrange(10))
num -= 1
return numbers
print(timeit.timeit("sum(generate(999))", setup="from __main__ import generate", number=1000))
>>> 0.88098192215 #Python 2.7
>>> 1.416813850402832 #Python 3.2
print(timeit.timeit("sum(create_list(999))", setup="from __main__ import create_list", number=1000))
>>> 0.924163103104 #Python 2.7
>>> 1.5026731491088867 #Python 3.2
```

这不仅是快了一点，也避免了你在内存中存储全部的列表!

缪斯的
翻译于 5
4人顶

顶 翻译得不错哦！

Ctypes的介绍

对于关键性的性能代码python本身也提供给我们一个API来调用C方法，主要通过 ctypes来实现，你可以不写任何C代码来利用ctypes。默认情况下python提供了预编译的标准c库，我们再回到生成器的例子，看看使用ctypes实现花费多少时间。

```
import timeit
from ctypes import cdll

def generate_c(num):
#Load standard C library
libc = cdll.LoadLibrary("libc.so.6") #Linux
```

缪斯的
翻译于 5
2人顶

顶 翻译得不错哦！

```
    #libc = cdll.msvcrt #Windows
    while num:
    yield libc.rand() % 10
    num -= 1

    print(timeit.timeit("sum(generate_c(999))", setup="from __main__ import generate_c", number=1000))
    >>> 0.434374809265 #Python 2.7
    >>> 0.7084300518035889 #Python 3.2
```

仅仅换成了c的随机函数，运行时间减了大半！现在如果我告诉你我们还能做得更好，你信吗?

Cython的介绍

缪斯的
翻译于 5
2人顶

顶 翻译得不错哦！

Cython 是python的一个超集，允许我们调用C函数以及声明变量来提高性能。尝试使用之前我们需要先安装Cython.

```
sudo pip install cython
```

Cython 本质上是另一个不再开发的类似类库Pyrex的分支，它将我们的类Python代码编译成C库，我们可以在一个python文件中调用。对于你的python文件使用.pyx后缀替代.py后缀，让我们看一下使用Cython如何来运行我们的生成器代码。

```
    #cython_generator.pyx
    import random

    def generate(num):
    while num:
    yield random.randrange(10)
    num -= 1
```

我们需要创建个setup.py以便我们能获取到Cython来编译我们的函数。

```
    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("generator", ["cython_generator.pyx"])]
    )
```

编译使用:

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

你应该可以看到两个文件cython_generator.c 文件 和 generator.so文件，我们使用下面方法测试我们的程序:

```
    import timeit
    print(timeit.timeit("sum(generator.generate(999))", setup="import generator", number=1000))
    >>> 0.835658073425
```

还不赖，让我们看看是否还有可以改进的地方。我们可以先声明“num”为整形，接着我们可以导入标准的C库来负责我们的随机函数。

```
    #cython_generator.pyx
    cdef extern from "stdlib.h":
    int c_libc_rand "rand"()

    def generate(int num):
    while num:
    yield c_libc_rand() % 10
    num -= 1
```

如果我们再次编译运行我们会看到这一串惊人的数字。

```
>>> 0.033586025238
```

仅仅的几个改变带来了不赖的结果。然而，有时这个改变很乏味，因此让我们来看看如何使用规则的python来实现吧。

其它翻译版本(1)

PyPy的介绍

PyPy 是一个Python2.7.3的即时编译器，通俗地说这意味着让你的代码运行的更快。Quora在生产环境中使用了PyPy。PyPy在它们的下载页面有一些安装说明，但是如果你使用的Ubuntu系统，你可以通过apt-get来安装。它的运行方式是立即可用的，因此没有疯狂的bash或者运行脚本，只需下载然后运行即可。让我们看看我们原始的生成器代码在PyPy下的性能如何。

缪斯的
翻译于 5
1人顶

顶 翻译得不错哦！

```
    import timeit
    import random

    def generate(num):
    while num:
    yield random.randrange(10)
    num -= 1

    def create_list(num):
    numbers = []
    while num:
    numbers.append(random.randrange(10))
    num -= 1
    return numbers
    print(timeit.timeit("sum(generate(999))", setup="from __main__ import generate", number=1000))
    >>> 0.115154981613 #PyPy 1.9
    >>> 0.118431091309 #PyPy 2.0b1
    print(timeit.timeit("sum(create_list(999))", setup="from __main__ import create_list", number=1000))
    >>> 0.140175104141 #PyPy 1.9
    >>> 0.140514850616 #PyPy 2.0b1
```

哇！没有修改一行代码运行速度是纯python实现的8倍。

进一步测试

crab
翻译于 5
1人顶

顶 翻译得不错哦！

为什么还要进一步研究？PyPy是冠军！并不全对。虽然大多数程序可以运行在PyPy上，但是还是有一些库没有被完全支持。而且，为你的项目写C的扩展相比换一个编译器更加容易。让我们更加深入一些，看看ctypes如何让我们使用C来写库。我们来测试一下归并排序和计算斐波那契数列的速度。下面是我们要用到的C代码（functions.c）：

```
/* functions.c */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/* http://rosettacode.org/wiki/Sorting_algorithms/Merge_sort#C */
inline void
merge (int *left, int l_len, int *right, int r_len, int *out)
{
  int i, j, k;
  for (i = j = k = 0; i < l_len && j < r_len;)
    out[k++] = left[i] < right[j] ? left[i++] : right[j++];
  while (i < l_len)
    out[k++] = left[i++];
  while (j < r_len)
    out[k++] = right[j++];
}

/* inner recursion of merge sort */
void
recur (int *buf, int *tmp, int len)
{
  int l = len / 2;
  if (len <= 1)
    return;
/* note that buf and tmp are swapped */
  recur (tmp, buf, l);
  recur (tmp + l, buf + l, len - l);
  merge (tmp, l, tmp + l, len - l, buf);
}

/* preparation work before recursion */
void
merge_sort (int *buf, int len)
{
/* call alloc, copy and free only once */
  int *tmp = malloc (sizeof (int) * len);
  memcpy (tmp, buf, sizeof (int) * len);
  recur (buf, tmp, len);
  free (tmp);
}

int
fibRec (int n)
{
  if (n < 2)
```

```
    return n;
  else
    return fibRec (n - 1) + fibRec (n - 2);
}
```

在Linux平台，我们可以用下面的方法把它编译成一个共享库：

```
gcc -Wall -fPIC -c functions.c
gcc -shared -o libfunctions.so functions.o
```

使用ctypes， 通过加载"libfunctions.so"这个共享库，就像我们前边对标准C库所作的那样，就可以使用这个库了。这里我们将要比较Python实现和C实现。现在我们开始计算斐波那契数列：

```
# functions.py

from ctypes import *
import time

libfunctions = cdll.LoadLibrary("./libfunctions.so")

def fibRec(n):
    if n < 2:
        return n
    else:
        return fibRec(n-1) + fibRec(n-2)

start = time.time()
fibRec(32)
finish = time.time()
print("Python: " + str(finish - start))

# C Fibonacci
start = time.time()
x = libfunctions.fibRec(32)
finish = time.time()
print("C: " + str(finish - start))
```

```
Python: 1.18783187866 #Python 2.7
Python: 1.272292137145996 #Python 3.2
Python: 0.563600063324 #PyPy 1.9
Python: 0.567229032516 #PyPy 2.0b1
C: 0.043830871582 #Python 2.7 + ctypes
C: 0.04574108123779297 #Python 3.2 + ctypes
C: 0.0481240749359 #PyPy 1.9 + ctypes
C: 0.046403169632 #PyPy 2.0b1 + ctypes
```

正如我们预料的那样，C比Python和PyPy更快。我们也可以用同样的方式比较归并排序。

gkgy
翻译于 5
1人顶

顶 翻译得不错哦！

我们还没有深挖Cypes库，所以这些例子并没有反映python强大的一面，Cypes库只有少量的标准类型限制，比如int型，char数组，float型，字节（bytes）等等。默认情况下，没有整形数组，然而通过与c_int相乘（ctype为int类型）我们可以间接获得这样的数组。这也是代码第7行所要呈现的。我们创建了一个c_int数组，有关我们数字的数组并分解打包到c_int数组中

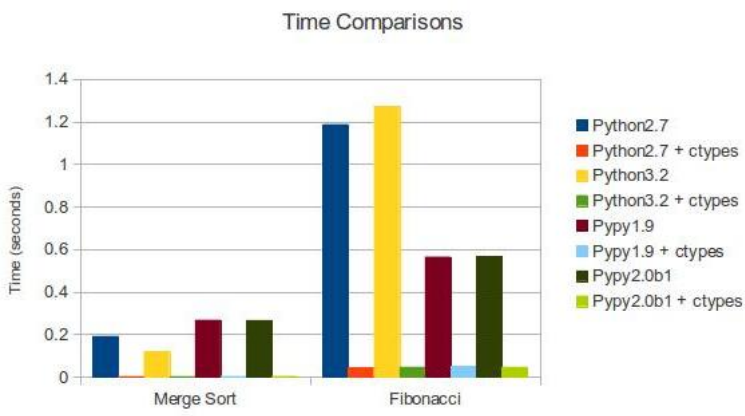
主要的是c语言不能这样做，而且你也不想。我们用指针来修改函数体。为了通过我们的c_numbers的数列，我们必须通过引用传递merge_sort功能。运行merge_sort后，我们利用c_numbers数组进行排序，我已经把下面的代码加到我的functions.py文件中了。

```
    #Python Merge Sort
    from random import shuffle, sample

    #Generate 9999 random numbers between 0 and 100000
    numbers = sample(range(100000), 9999)
    shuffle(numbers)
    c_numbers = (c_int * len(numbers))(*numbers)

    from heapq import merge
    def merge_sort(m):
    if len(m) <= 1:
    return m
    middle = len(m) // 2
    left = m[:middle]
    right = m[middle:]
    left = merge_sort(left)
    right = merge_sort(right)
    return list(merge(left, right))

    start = time.time()
    numbers = merge_sort(numbers)
    finish = time.time()
```

```
    print("Python: " + str(finish - start))

    #C Merge Sort
    start = time.time()
    libfunctions.merge_sort(byref(c_numbers), len(numbers))
    finish = time.time()
    print("C: " + str(finish - start))
```

```
Python: 0.190635919571 #Python 2.7
Python: 0.11785483360290527 #Python 3.2
Python: 0.266992092133 #PyPy 1.9
Python: 0.265724897385 #PyPy 2.0b1
C: 0.00201296806335 #Python 2.7 + ctypes
C: 0.0019741058349609375 #Python 3.2 + ctypes
C: 0.0029308795929 #PyPy 1.9 + ctypes
C: 0.00287103652954 #PyPy 2.0b1 + ctypes
```

这儿通过表格和图标来比较不同的结果。



	Merge Sort	Fibonacci
Python 2.7	0.191	1.187
Python 2.7 + ctypes	0.002	0.044
Python 3.2	0.118	1.272
Python 3.2 + ctypes	0.002	0.046
PyPy 1.9	0.267	0.564
PyPy 1.9 + ctypes	0.003	0.048
PyPy 2.0b1	0.266	0.567
PyPy 2.0b1 + ctypes	0.003	0.046

希望你利用C和PyPy优化你的python代码并以此为敲门砖找到一个好职位。像往常一样如果你有任何意见或问题，请随时把评论下载下面或者在我的网页上与我取得联系。感谢您的阅读！

附：如果您的公司正在寻求聘请即将毕业的优秀大学生（2013年5月），让我知道！

本文中的所有译文仅用于学习和交流目的，转载请务必注明文章译者、出处、和本文链接
我们的翻译工作遵照 CC 协议，如果我们的工作有侵犯到您的权益，请及时联系我们

评论(18)

Ctrl/CMD+Enter 发表评论

crab2313 发表于 2013-03-24 00:26

@gkgy 哥们， 没用过ctypes？

crab2313 发表于 2013-03-24 00:54

上面几位有闲工夫能把代码格式一下么， 空格全掉了

renwofei423 发表于 2013-03-24 09:57

能参考原文整理下格式吗？
http://maxburstein.com/blog/speeding-up-your-python-code/
@缪斯的情人 @gkgy

首席安全砖家 发表于 2013-03-24 12:02

> 引用来自“renwofei423”的评论
> 能参考原文整理下格式吗？
> http://maxburstein.com/blog/speeding-up-your-python-code/
> @缪斯的情人 @gkgy

这个网站显示代码的，用的什么插件？

优雅先生 发表于 2013-03-24 12:03

格式

renwofei423 发表于 2013-03-24 12:11

> 引用来自“钟晓骏”的评论
> > 引用来自“renwofei423”的评论
> > 能参考原文整理下格式吗？
> > http://maxburstein.com/blog/speeding-up-your-python-code/
> > @缪斯的情人 @gkgy
>
> 这个网站显示代码的，用的什么插件？

sorry 我也搞不清。。。

缪斯的情人 发表于 2013-03-24 16:18

> 引用来自“renwofei423”的评论
> 能参考原文整理下格式吗？
> http://maxburstein.com/blog/speeding-up-your-python-code/
> @缪斯的情人 @gkgy

这玩意不是我管得了的，找@红薯 @小编辑 吧

星塵子 发表于 2013-03-24 16:53

> 引用来自“钟晓骏”的评论
> > 引用来自“renwofei423”的评论
> > 能参考原文整理下格式吗？
> > http://maxburstein.com/blog/speeding-up-your-python-code/
> > @缪斯的情人 @gkgy
>
> 这个网站显示代码的，用的什么插件？

代码插件：Google Code prettify 。详见： http://goo.gl/7WLJf

首席安全砖家 发表于 2013-03-24 16:56

> 引用来自“星塵子”的评论
> > 引用来自“钟晓骏”的评论
> > > 引用来自“renwofei423”的评论
> > > 能参考原文整理下格式吗？
> > > http://maxburstein.com/blog/speeding-up-your-python-code/
> > > @缪斯的情人 @gkgy

> 这个网站显示代码的，用的什么插件？
>
> 代码插件：Google Code prettify 。详见： http://goo.gl/7WLJf

谢谢， 这种格式的代码插件正和我意，It 's so cool .

晒太阳的小猪 发表于 2013-03-24 19:48
呵呵，很棒哦！！

战争总会来临 发表于 2013-03-25 13:58
pretty Good.

crf1111 发表于 2013-03-26 09:00
very good

love-Teddy 发表于 2013-03-26 13:28
写的很好!

TankyWoo 发表于 2013-03-26 22:24

> 引用来自“星塵子”的评论
>
> > 引用来自“钟晓骏”的评论
> >
> > > 引用来自“renwofei423”的评论
> > > 能参考原文整理下格式吗？
> > > http://maxburstein.com/blog/speeding-up-your-python-code/
> > > @缪斯的情人 @gkgy
> >
> > 这个网站显示代码的，用的什么插件？
>
> 代码插件：Google Code prettify 。详见： http://goo.gl/7WLJf

这明明是SyntaxHighlighter好不好！官网：http://alexgorbatchev.com/SyntaxHighlighter/， oschina上介绍：http://www.oschina.net/p/syntaxhighlighter

TankyWoo 发表于 2013-03-26 22:24

> 引用来自“钟晓骏”的评论
>
> > 引用来自“星塵子”的评论
> >
> > > 引用来自“钟晓骏”的评论
> > >
> > > > 引用来自“renwofei423”的评论
> > > > 能参考原文整理下格式吗？
> > > > http://maxburstein.com/blog/speeding-up-your-python-code/
> > > > @缪斯的情人 @gkgy
> > >
> > > 这个网站显示代码的，用的什么插件？
> >
> > 代码插件：Google Code prettify 。详见： http://goo.gl/7WLJf
>
> 谢谢， 这种格式的代码插件正和我意，It 's so cool .

见我楼上回复

首席安全砖家 发表于 2013-03-26 22:33

> 引用来自“TankyWoo”的评论
>
> > 引用来自“钟晓骏”的评论
> >
> > > 引用来自“星塵子”的评论
> > >
> > > > 引用来自“钟晓骏”的评论
> > > >
> > > > > 引用来自“renwofei423”的评论
> > > > > 能参考原文整理下格式吗？
> > > > > http://maxburstein.com/blog/speeding-up-your-python-code/
> > > > > @缪斯的情人 @gkgy
> > > >
> > > > 这个网站显示代码的，用的什么插件？
> > >
> > > 代码插件：Google Code prettify 。详见： http://goo.gl/7WLJf
> >
> > 谢谢， 这种格式的代码插件正和我意，It 's so cool .
>
> 见我楼上回复

换了，我说的是原文的http://maxburstein.com/blog/speeding-up-your-python-code/

hssdx 发表于 2013-03-27 16:18

> 引用来自“renwofei423”的评论
>
> > 引用来自“钟晓骏”的评论
> >
> > > 引用来自“renwofei423”的评论
> > > 能参考原文整理下格式吗？
> > > http://maxburstein.com/blog/speeding-up-your-python-code/
> > > @缪斯的情人 @gkgy
> >
> > 这个网站显示代码的，用的什么插件？
>
> sorry 我也搞不清。。。

网上挺多这些插件把

Tmac 发表于 2013-03-28 09:44

翻译的不错