## cProfiler对python程序进行可视化性能分析

## 前言

当我们需要对python程序进行优化时，第一步要做的并不是盲目去优化，而是首先要对我们现有的程序进行分析，发现程序的性能瓶颈然后进行针对性的优化，这里采用Python中常用的性能分析器cProfiler，并使用Gprof2Dot将分析器输出转换成Graphviz可处理的图像表述，配合dot命令，即可得到不同函数所消耗的时间分析图。

## 正文

第一步

我们需要写一个带有参数的性能分析装饰器，其中主要用到cProfile模块的Profile类和pstat模块的Stats类。

* Profile类：
* enable(): 开始收集性能分析数据
* disable(): 停止收集性能分析数据
* create\_stats(): 停止收集分析数据，并为已收集的数据创建stats对象
* print\_stats(): 创建stats对象并打印分析结果
* dump\_stats(filename): 把当前性能分析的结果写入文件(二进制格式)
* runcall(func, \*args, \*\*kwargs): 收集被调用函数func的性能分析数据
* Stats类：
* pstats模块提供的Stats类可以帮助我们读取和操作stats文件（二进制格式）

下面直接上代码

import cProfile

import pstats

import os

# 性能分析装饰器定义

def do\_cprofile(filename):

def wrapper(func):

def profiled\_func(\*args, \*\*kwargs):

# Flag for do profiling or not.

DO\_PROF = os.getenv("PROFILING")

if DO\_PROF:

profile = cProfile.Profile()

profile.enable()

result = func(\*args, \*\*kwargs)

profile.disable()

# Sort stat by internal time.

sortby = "tottime"

ps = pstats.Stats(profile).sort\_stats(sortby)

ps.dump\_stats(filename)

else:

result = func(\*args, \*\*kwargs)

return result

return profiled\_func

return wrapper

这样我们就可以在我们想进行分析的地方进行性能分析，例如我想分析supplier.py中class SupplierViewSet(BaseViewSet):类中的 get\_queryset()方法

class SupplierViewSet(BaseViewSet):

# ...

# 应用装饰器来分析函数

@do\_cprofile("./cpf\_run.prof")

def get\_queryset(self):

# ...

### 第二步

装饰器函数中通过sys.getenv来获取环境变量判断是否需要进行分析，因此可以通过设置环境变量来告诉程序是否进行性能分析:

export PROFILING=y

程序跑完后便会在当前路径下生成cpf\_run.prof的分析文件，此时我们需要将该文件从远程服务器取到本地，然后我们就可以通过可视化工具对函数进行分析。

### 第三步

下载gprof2dot.py 项目地址：<https://github.com/jrfonseca/gprof2dot>

下载Graphviz工具

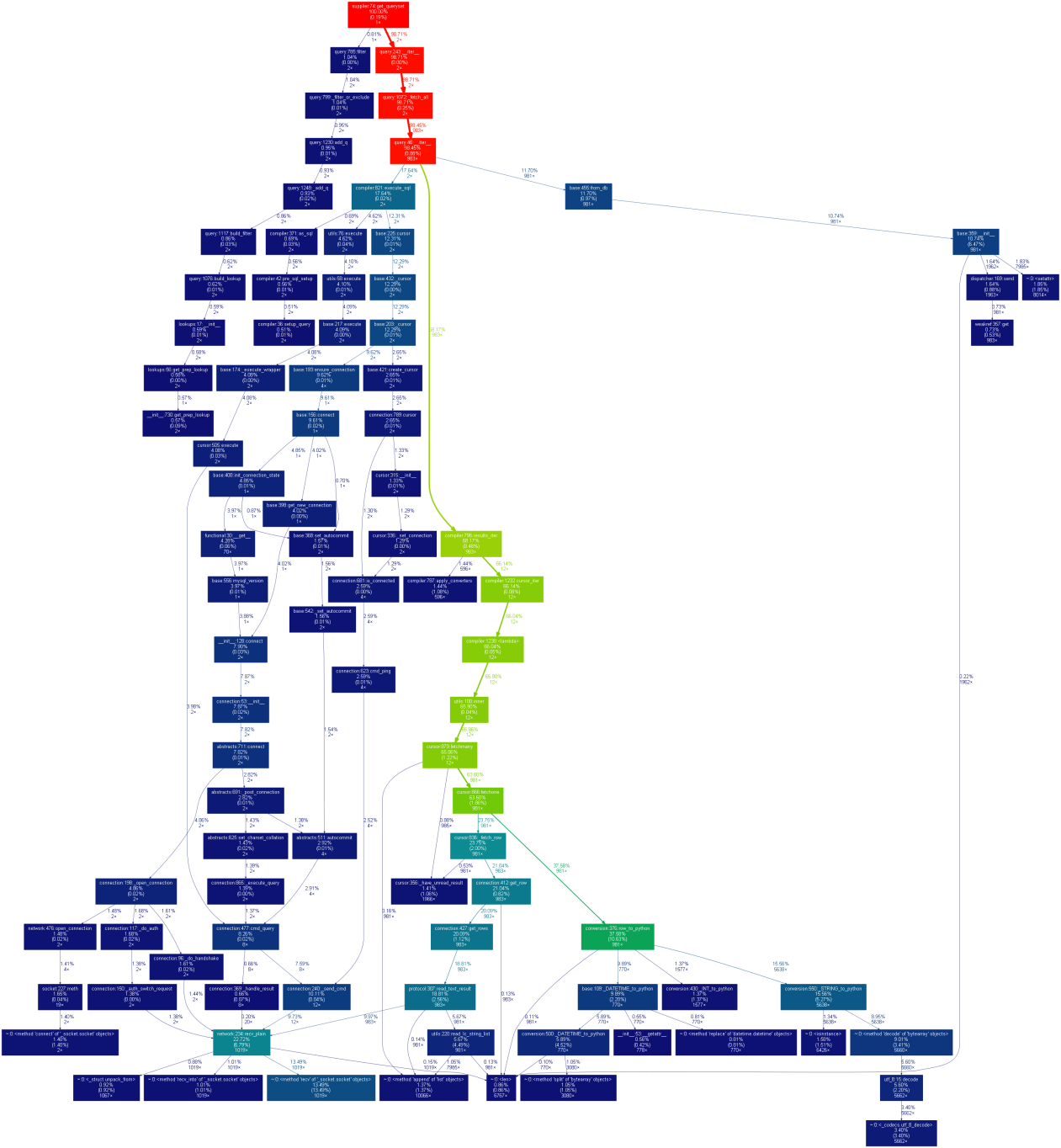
将gprof2dot.py python文件放入Graphviz的bin目录下

将生成的cpf\_run.prof文件放入Graphviz的bin目录下

然后在bin路径下

python gprof2dot.py -f pstats cpf\_run.prof | dot -Tpng -o cpf\_run.png

生成cpf\_run.png



### 第四步

根据可视化图片进行分析，顺着浅色方格的看下去很容易发现程序的瓶颈部分

每个node的信息如下:

+------------------------------+|

| function name |

| total time % ( self time % ) |

| total calls |

+------------------------------+

每个edge的信息如下:

total time %

calls

parent --------------------> children

具体含义可以在<https://github.com/jrfonseca/gprof2dot>中看到，这里只做部分解释。