## **vn.py社区精选17 - 策略回测性能分析入门**

原创 用Python的交易员 VNPY  2020-01-26

收录于话题

#vn.py社区精选

25个

原文作者：张国平

**cProfile介绍**

较为大型的程序在开发完成后，通常都会采用以下步骤来进行性能优化：

对代码的执行效能测量与分析（profiling）；

找出程序执行耗时中的瓶颈所在；

针对少数关键的热点代码进行优化改进；

提高整体程序的执行速度。

在Python中如果要进行计算机程序执行效率的分析，有好几种方式可以使用（具体请参考Python官方手册），而对于大部分用户来说，最常用的方法就是使用cProfile模块。

cProfile模块是一个Python标准库内建的分析工具，采用hook进CPython虚拟机的方式，来测量每一个函数运行所花费的时间。尽管hook本身会导致显著的额外开销，但通过这种测量方式我们能获得更多的细节信息，有些甚至能帮助我们在代码的底层执行细节中找到令人惊讶的发现。

先来了解下cProfile的输出信息，如下图所示的内容整体可以分成6大类：

ncalls：表示函数调用的次数；

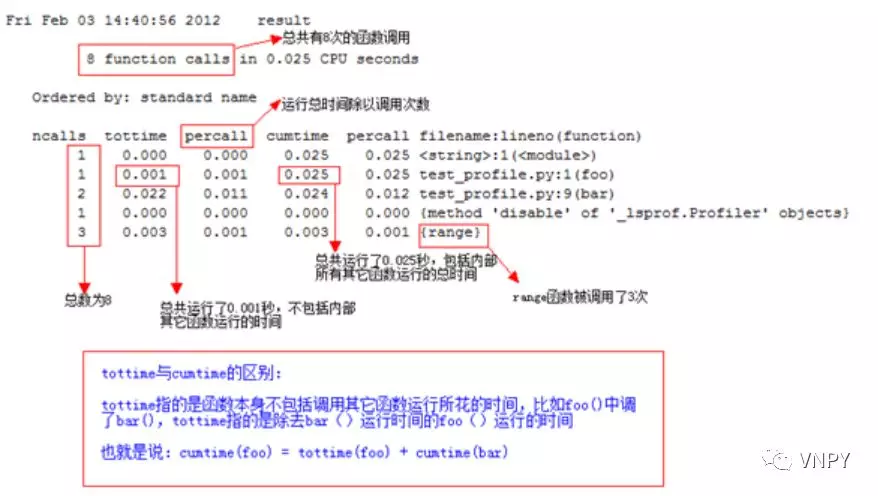
tottime：表示指定函数的总的运行时间，除掉函数中调用子函数的运行时间；

percall：（第一个percall）等于 tottime/ncalls；

cumtime：表示该函数及其所有子函数的调用运行的时间，即函数开始调用到返回的时间；

percall：（第二个percall）即函数运行一次的平均时间，等于 cumtime/ncalls；

filename：lineno(function)：每个函数调用的具体信息



**使用Jupyter Notebook执行分析**

第一步是加载相关模块，注意cProfile是Python内置标准库，直接import即可：

from datetime import datetimeimport cProfile  
from vnpy.app.cta\_strategy.backtesting import BacktestingEngine, OptimizationSettingfrom vnpy.app.cta\_strategy.strategies.atr\_rsi\_strategy import AtrRsiStrategy

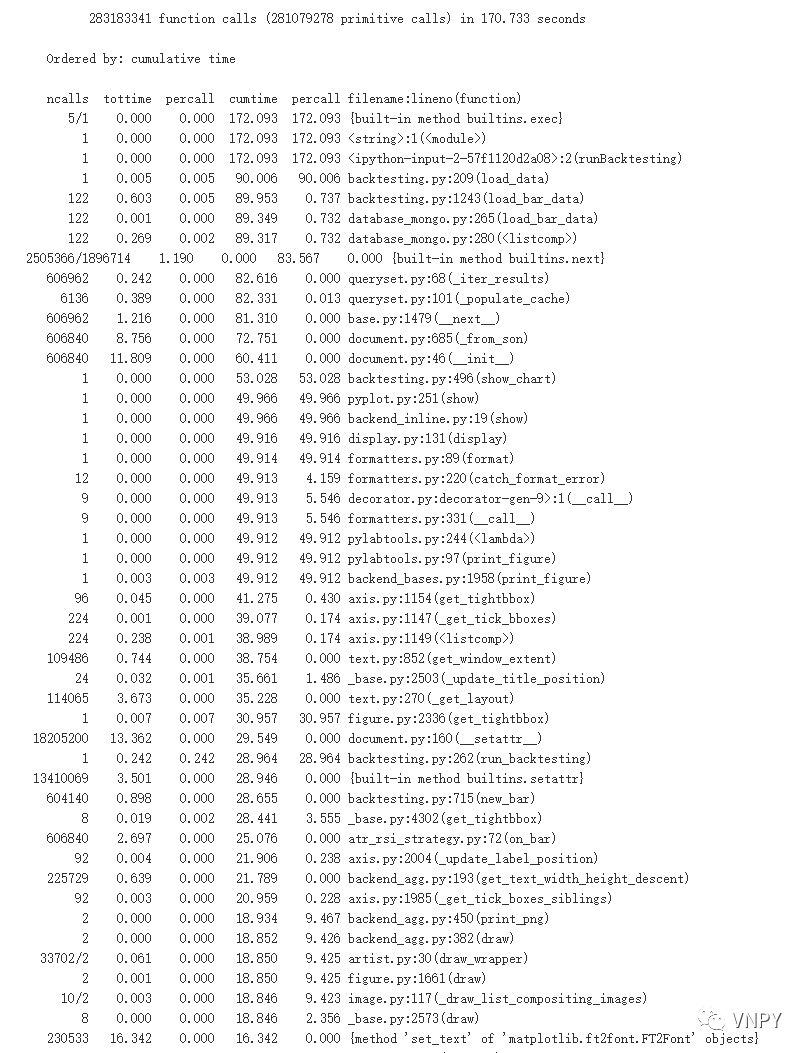
第二步，定义回测函数，里面需要填入策略回测品种所对应的参数：

#%%def runBacktesting(): engine = BacktestingEngine() engine.set\_parameters( vt\_symbol="IF888.CFFEX", interval="1m", start=datetime(2010, 1, 1), end=datetime(2019, 12, 30), rate=0.3/10000, slippage=0.2, size=300, pricetick=0.2, capital=1\_000\_000,    )    engine.add\_strategy(AtrRsiStrategy, {}) engine.load\_data() engine.run\_backtesting()    engine.calculate\_result() engine.calculate\_statistics() engine.show\_chart()

最后，使用cProfile来运行runBacktesting函数，并将输出的内容按照cumtime进行排序：

cProfile.run("runBacktesting()", sort="cumulative")

性能分析的输出结果如下图所示：



**基于gprof2dot的可视化展示**

尽管我们已经对输出的结果进行了排序处理，但肉眼看起来还是不够直观。比起密密麻麻看得眼睛都疼的一堆数字，还是可视化图表的呈现效果更为清晰，所幸在Python性能分析这块，同样已经有了成熟的工具体系可以实现。

**安装graphvi**

graphvi最方便的地方在于能够很快的清晰的画出点与点之间的关系，并且有许多布局算法能够很好的去布局，安装流程如下：

进入graphvi官网（<https://www.graphviz.org/>），如果是windows系统，找到对应的msi程序并且点击下载（<https://graphviz.gitlab.io/_pages/Download/windows/graphviz-2.38.msi>）；

安装msi程序，安装完毕后打开文件夹C:\Program Files (x86)\Graphviz2.38\bin，保证能看到文件【dot.ext】；

在环境变量path中，加入【C:\Program Files (x86)\Graphviz2.38\bin】，保存退出

**安装gprof2dot**

执行下面代码，安装gprof2dot。

pip install gprof2dot

**分析生成可视化图表**

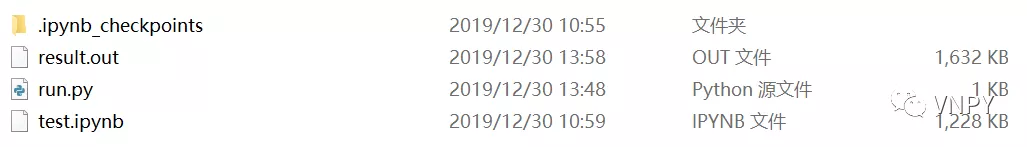
首先需要创建策略回测用的脚本文件run.py，里面的内容整体上和我们在jupyter notebook中使用的回测逻辑类似：

from datetime import datetime  
from vnpy.app.cta\_strategy.backtesting import BacktestingEngine, OptimizationSettingfrom vnpy.app.cta\_strategy.strategies.atr\_rsi\_strategy import AtrRsiStrategy  
def runBacktesting(): engine = BacktestingEngine() engine.set\_parameters( vt\_symbol="IF888.CFFEX", interval="1m", start=datetime(2010, 1, 1), end=datetime(2019, 12, 30), rate=0.3/10000, slippage=0.2, size=300, pricetick=0.2, capital=1\_000\_000, ) engine.add\_strategy(AtrRsiStrategy, {}) engine.load\_data() engine.run\_backtesting() engine.calculate\_result() engine.calculate\_statistics() engine.show\_chart()  
if \_\_name\_\_ == '\_\_main\_\_': runBacktesting()

然后在该文件夹下，按住Shift键点击鼠标右键，选择【在此处打开命令窗口】或者【在此处打开PowerShell】进入命令提示符环境，并运行cProfile进行性能测试：

python -m cProfile -o result.out run.py

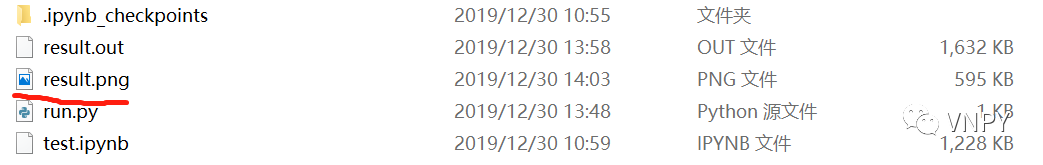
回测完毕后，会将性能分析输出的结果自动写入到result.out文件中，注意该文件是二进制的格式，无法直接查看：



回到命令提示符中（cmd或者Powershell），运行下列命令来使用gpro2dot把result.out转换成png图表：

python -m gprof2dot -f pstats result.out | dot -Tpng -o result.png

执行完毕后，我们可以看到文件夹内新增了png文件，如下图所示：



打开这个png图片，我们可以清晰地看到每段代码的执行顺序，以及它所消耗的时长比例，这样更易于我们找到不必要的代码开销，从而实现性能优化：



除了本文中使用的分析案例CTA回测功能外，cProfile模块同样可以用于vn.py整体性能方面的优化改进，通过上述流程找出热点函数代码段后，针对其采用更高效的数据容器、改进逻辑算法细节或者使用Numba和Cython等技术来提升数学计算速度等等。