我的Python进程怎么了

Python进程调试和监控

张翔

CONTENTS | 目录

为什么我的Python进程卡住了?

为什么我的Python进程消耗这么多的内存?

为什么我的Python进程消耗了这么多CPU?

print & log

优点:
通常很有用

缺点:
需要了解你的代码
需要添加、删除、重启

pdb

```
python3 -m pdb myscript.py
```

```
import pdb; pdb.set_trace()
```

```
breakpoint() (new in 3.7)
```

sys

```
sys.excepthook
sys.getallocatedblocks
sys.setprofile
sys.settrace
sys.set_asyncgen_hooks
sys.set_coroutine_wrapper
sys._current_frames
sys._getframe
more …
```

```
import sys, traceback

for frames in sys._current_frames().values():
    traceback.print_stack(frames)

def print_stack()
    traceback.print_stack(sys._getframe(1))
```

gc

```
gc.get_objects
gc.get_referrers
gc.get_referents

import gc, greenlet, traceback
for obj in gc.get_objects():
    if instance(obj, greenlet.greenlet):
        traceback.print_stack(obj.gr_frame)
```

tracemalloc

```
import tracemalloc
tracemalloc.start()
# ... run your application ...
snapshot = tracemalloc.take_snapshot()
top_stats = snapshot.statistics('lineno')
print("[ Top 10 ]")
for stat in top_stats[:10]:
    print(stat)
```

tracemalloc

```
import tracemalloc
tracemalloc.start()
# ... start your application ...
snapshot1 = tracemalloc.take_snapshot()
# ... call the function leaking memory ...
snapshot2 = tracemalloc.take_snapshot()
top_stats = snapshot2.compare_to(snapshot1, 'lineno')
print("[ Top 10 differences ]")
for stat in top_stats[:10]:
    print(stat)
```

more: trace, faulthandler

优点:

解释器自带

完善的文档和社区支持

功能更强大

cons:

需要添加、删除、重启

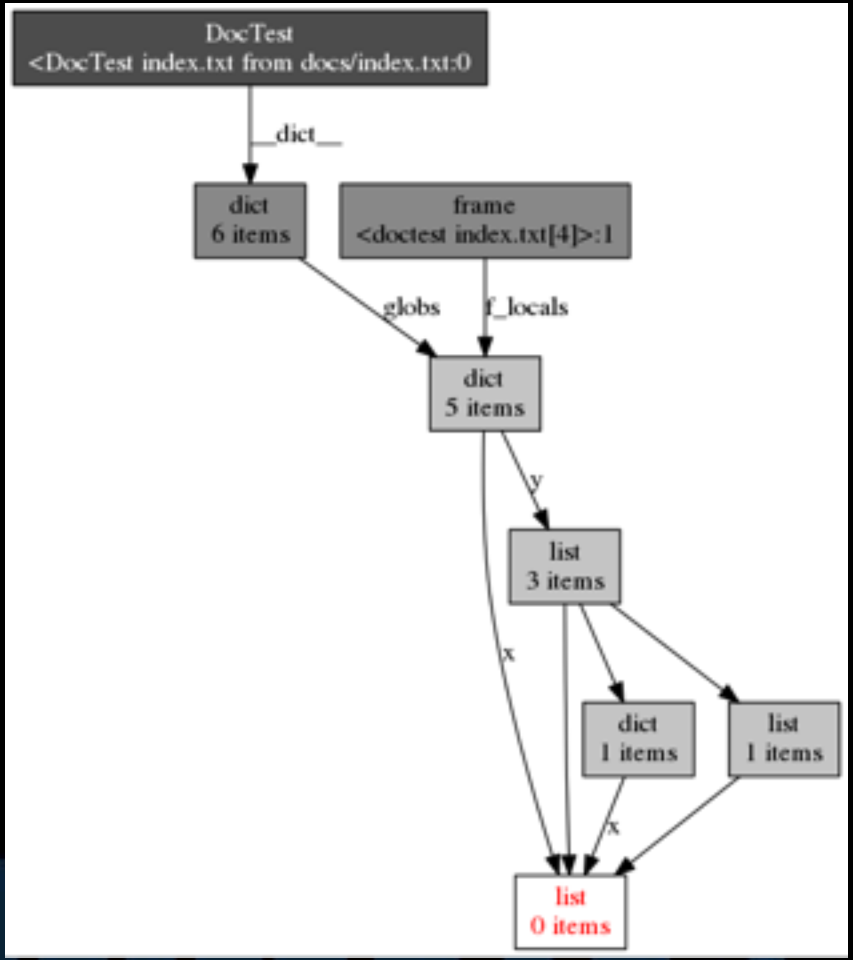
heapy

不支持C extension

```
>>> from guppy import hpy;
>>> hpy().heap()
Partition of a set of 48477 objects. Total size = 3265516 bytes.
 Index  Count   %     Size   % Cumulative  % Kind (class / dict of class)
     0  25773  53  1612820  49   1612820  49 str
     1  11699  24   483960  15   2096780  64 tuple
     2    174   0   241584   7   2338364  72 dict of module
     3   3478   7   222592   7   2560956  78 types.CodeType
     4   3296   7   184576   6   2745532  84 function
     5    401   1   175112   5   2920644  89 dict of class
     6    108   0    81888   3   3002532  92 dict (no owner)
     7    114   0    79632   2   3082164  94 dict of type
     8    117   0    51336   2   3133500  96 type
     9    667   1    24012   1   3157512  97 __builtin__.wrapper_descriptor
<76 more rows. Type e.g. '_.more' to view.>
```

objgraph

```
>>> objgraph.show_most_common_types()
tuple                      5224
function                   1329
wrapper_descriptor         967
dict                       790
builtin_function_or_method 658
method_descriptor          340
weakref                    322
list                       168
member_descriptor          167
type                       163
```

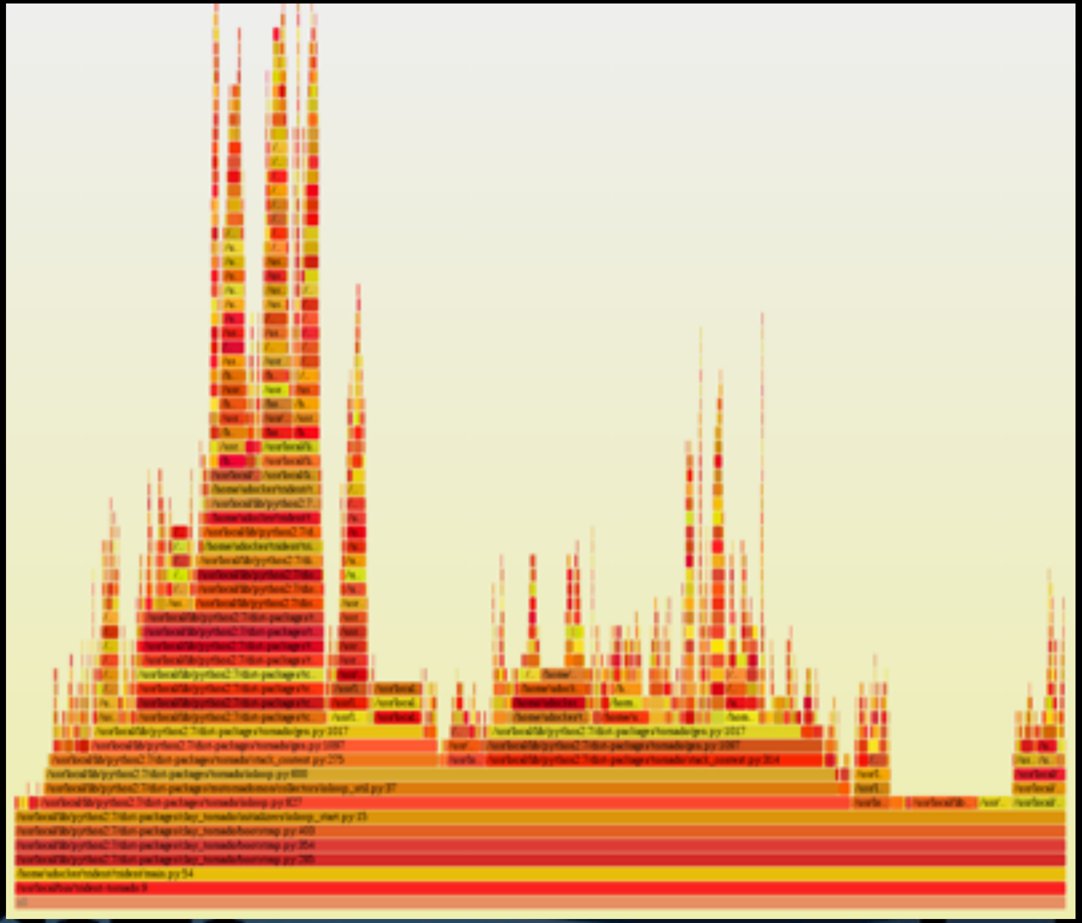


more: meliae, pysizer, memory_profiler …

优点:
功能强大
使用简单
缺点:
需要添加、删除、重启
文档和社区支持并不完善
有时候并不能如预料的工作

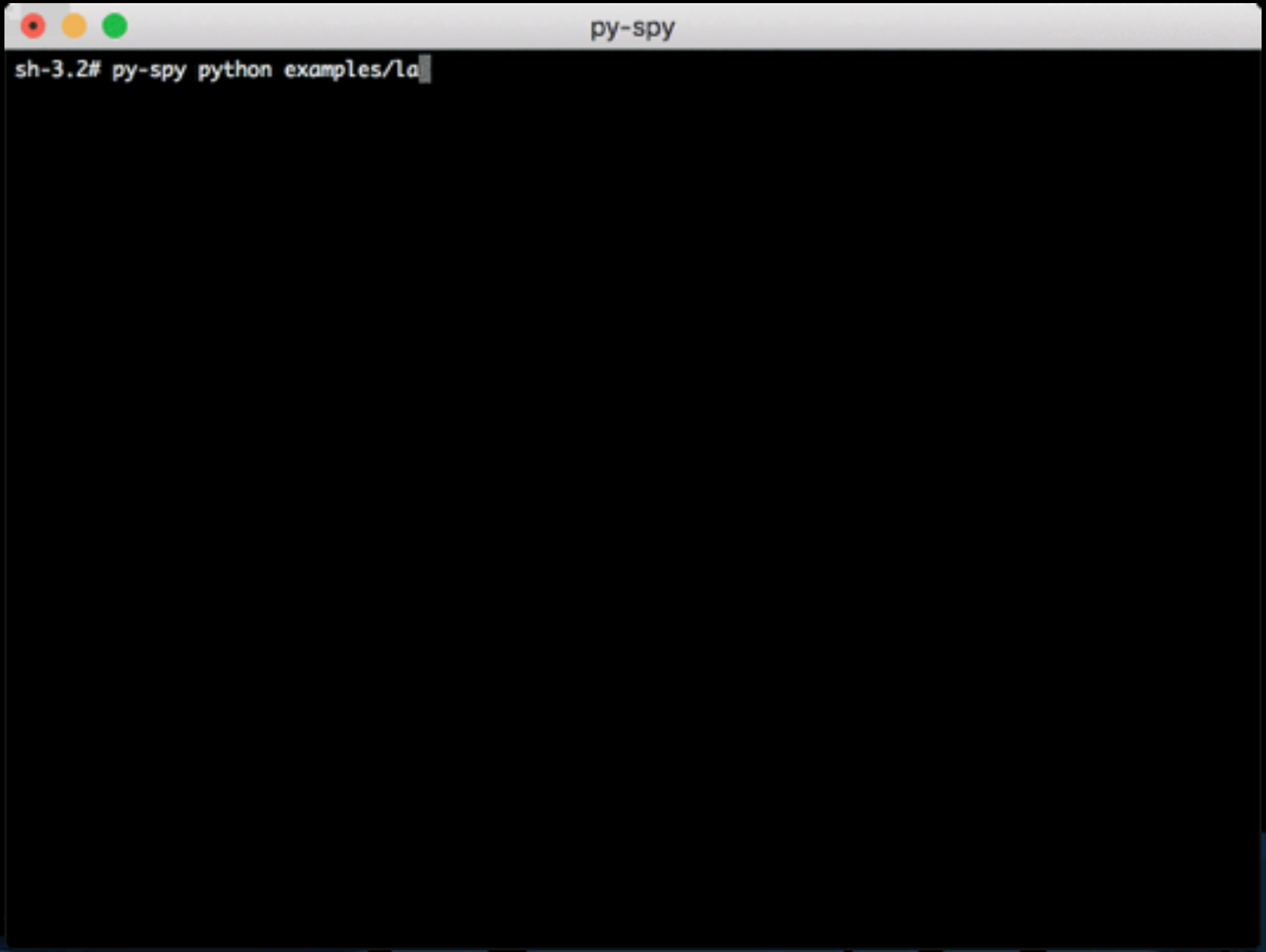
pyflame

```
# Generate flame graph for pid 12345; assumes flamegraph.pl is in your $PATH.
pyflame -s 60 -r 0.01 —p 12345 | flamegraph.pl > myprofile.svg
```



py-spy

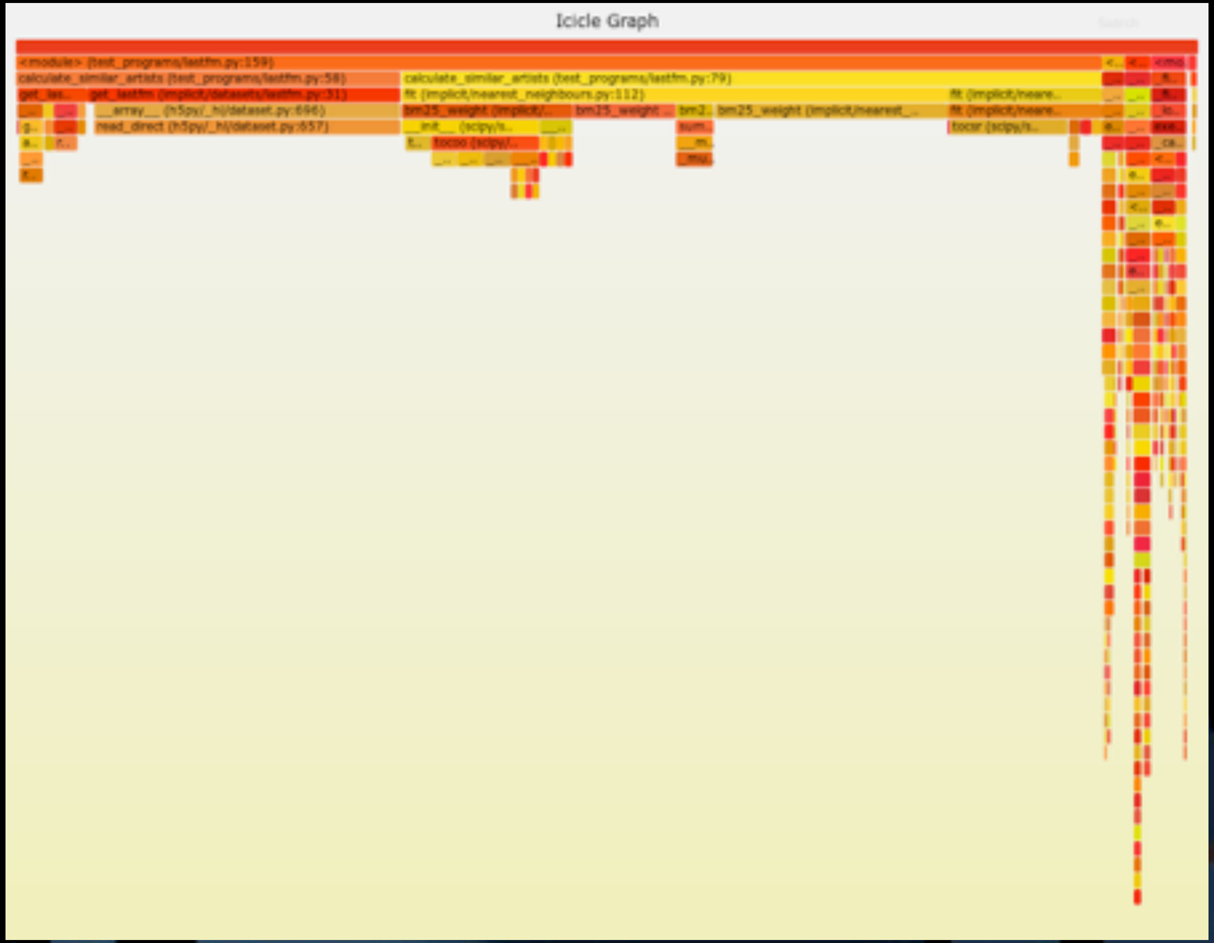
```
py-spy --pid 12345
```



```
sh-3.2# py-spy python examples/la
```

py-spy

```
py-spy --flame profile.svg --pid 12345
```



这里的功能都可以以子进程的方式启动，但我不常用

pyrasite

A interactive gui for memory viewer
use meliae underneath
has a pyrasite-gui

shell可以和之前的工具配置达到不重启

```
pyrasite-shell 12345

Pyrasite Shell 2.0
Connect to ‘python /tmp/test.py’
Python 2.7.5 (default, Jun 17 2018, 12:46:58)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-28)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
(DistantInteractiveConsole)
>>> import sys
>>> sys._current_frames()
{4548400576: <frame object at 0x1099701f8>, 4656784832: <frame object at 0x10c0081f8>}

pyrasite-memory-viewer 12345
```

pydevd

默认需要一个remote debug server来做交互。

```
python attach_pydevd.py —pid 12345

Attaching with arch: i386:x86-64
Running: gdb —nw —nh —nx —pid 12345 —batch —eval-command=‘set scheduler-locking
Off’ —eval-command=‘set architecture i386:x86-64’ —eval-command=‘call dlopen(
“/usr/pydevd_attach_to_process/attach_linux_amd64.so”, 2)’ —eval-command=‘call
DoAttach(0, “import sys; sys.path.append(\”\”);sys.path.append(\”/usr/pydevd_
attach_to_process\”);import attach_script;attach_script.attach(port=5678, host=
\”127.0.0.1\”);”, 0)’ —command=‘/usr/pydevd_attach_to_process/linux/gdb_threads_
settrace.py’
```

more: pyringe, pytools …

优点:

不需要添加、删除、重启

功能强大，使用简单

缺点:

flamegraph对递归的程序展示不好

调用栈并不能展示C stack

并不能跨解释器

依赖操作系统的配置

循环的情况可以使用callgrind格式

PYCON CHINA 2018

gdb

```
gdb python 12345
```

```
(gdb) bt
#0  0x0000002a95b3b705 in raise () from /lib/libc.so.6
#1  0x0000002a95b3ce8e in abort () from /lib/libc.so.6
#2  0x00000000004c164f in posix_abort (self=0x0, noargs=0x0)
    at ../Modules/posixmodule.c:7158
#3  0x0000000000489fac in call_function (pp_stack=0x7fbffff110, oparg=0)
    at ../Python/ceval.c:3531
#4  0x0000000000485fc2 in PyEval_EvalFrame (f=0x66ccd8)
    at ../Python/ceval.c:2163
...
```

gdb

```
(gdb) py-list
2025        # Open external files with our Mac app
2026        if sys.platform == "darwin" and 'Spyder.app' in __file__:
2027            main.connect(app, SIGNAL('open_external_file(QString)'),
2028                         lambda fname: main.open_external_file(fname))
2029
>2030        app.exec_()
2031        return main
2032
2033
2034    def __remove_temp_session():
2035        if osp.isfile(TEMP_SESSION_PATH):
```

py-bt py-up py-down
对所有线程应用

dtrace/systemtap (new in 3.6)

```
probe process("python").mark("function__entry") {
     filename = user_string($arg1);
     funcname = user_string($arg2);
     lineno = $arg3;

     printf("%s => %s in %s:%d\\n",
            thread_indent(1), funcname, filename, lineno);
}

probe process("python").mark("function__return") {
    filename = user_string($arg1);
    funcname = user_string($arg2);
    lineno = $arg3;

    printf("%s <= %s in %s:%d\\n",
           thread_indent(-1), funcname, filename, lineno);
}
```

内置的支持需要在编译的时候开启，可以用dtrace或者readelf来查看。

dtrace/systemtap (new in 3.6)

more markers, gc, import, line

```
stap show-call-hierarchy.stp -c "./python test.py"
```

```
11408 python(8274):        => __contains__ in Lib/_abcoll.py:362
11414 python(8274):         => __getitem__ in Lib/os.py:425
11418 python(8274):          => encode in Lib/os.py:490
11424 python(8274):          <= encode in Lib/os.py:493
11428 python(8274):         <= __getitem__ in Lib/os.py:426
11433 python(8274):        <= __contains__ in Lib/_abcoll.py:366
```

perf, bcc特别关于Python的是可以容易的看内存的申请、释放

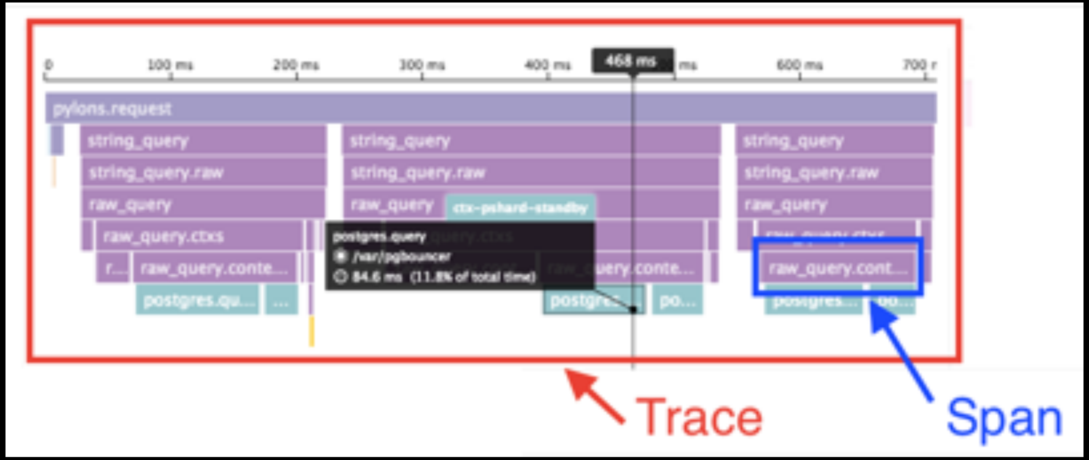
more: perf, bcc, tcpdump, dstat ...

优点:
不需要添加、删除、重启
功能强大

缺点:
并不能跨解释器、跨OS
需要更多的学习
依赖于操作系统的配置

APM

```
from datadog import initialize
options = {
    'api_key':'<DATADOG_API_KEY>',
    'app_key':'<DATADOG_APP_KEY>'
}
initialize(**options)
# Use Datadog REST API client
from datadog import api
title = "Something big happened!"
text = 'And let me tell you all about it here!'
tags = ['version:1', 'application:web']
api.Event.create(title=title, text=text, tags=tags)
```



带分布式追踪，告警，漂亮的dashboard，可以与其他语言结合

输出的信息多是应用、框架的，HTTP，中间件，SQL，Exception

技术就是monkey-patch

what’s possible

动态加载解决后悔的问题

debugging解决开发的问题，跨解释器

builtin dynamic attaching mechanism

```
python -m runtime —target 12345 -c “import traceback;traceback.print_stack()”

pdb —pid 12345

py-stack 12345
```

official builtin debugging infrastructure

THANK YOU

Email: angwerzx@126.com