Tracing, Profiling & Debugging in Production

Trent Lloyd

twitter.com/lathiat

Tracing, Profiling & Debugging in Production

Trent Lloyd

twitter.com/lathiat

Tracing - Precisely logging **every** invocation of a **line** or **function**

Profiling - **Sampling** a program's currently executing **line** or **function** at a set interval

Debugging - ~~**Interactively** inspecting a program's execution state~~ ***or*** diagnosing and fixing a bug

Profiling

What is my Python program doing?

Stack Traces

Where are we?

How do we get a stack trace?

```
traceback.print_stack()
```

In-process profiling

cProfile

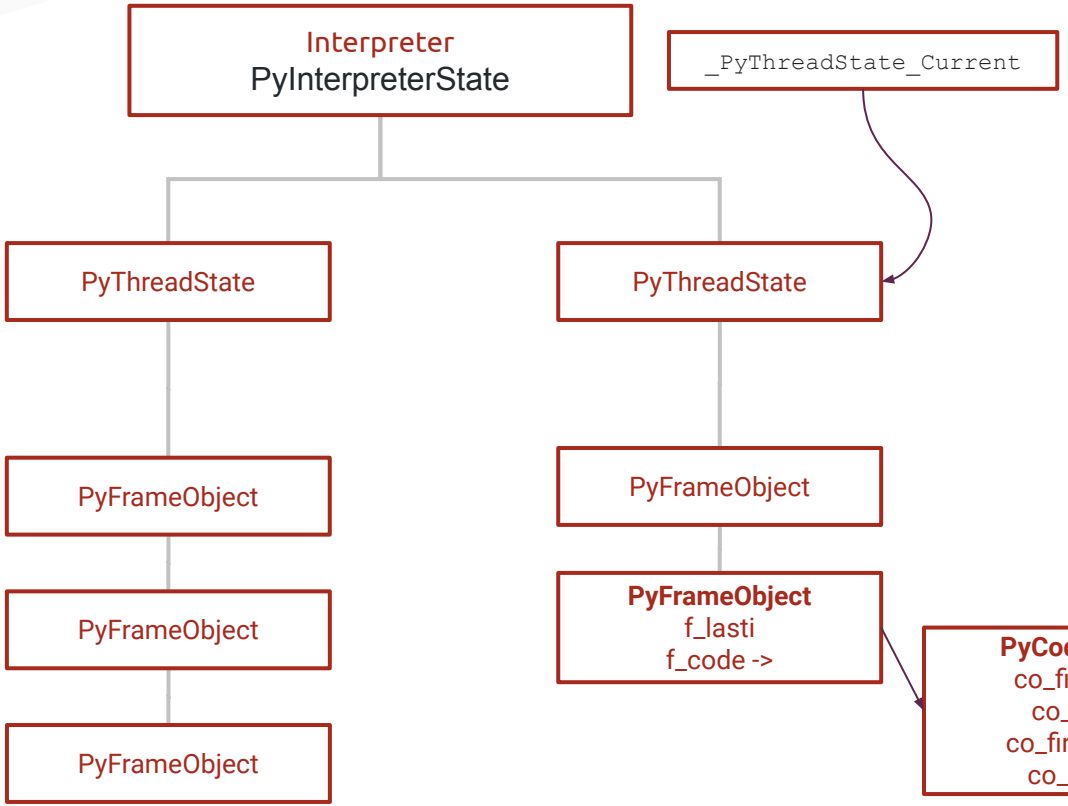
profile

pyinstrument

vmprof

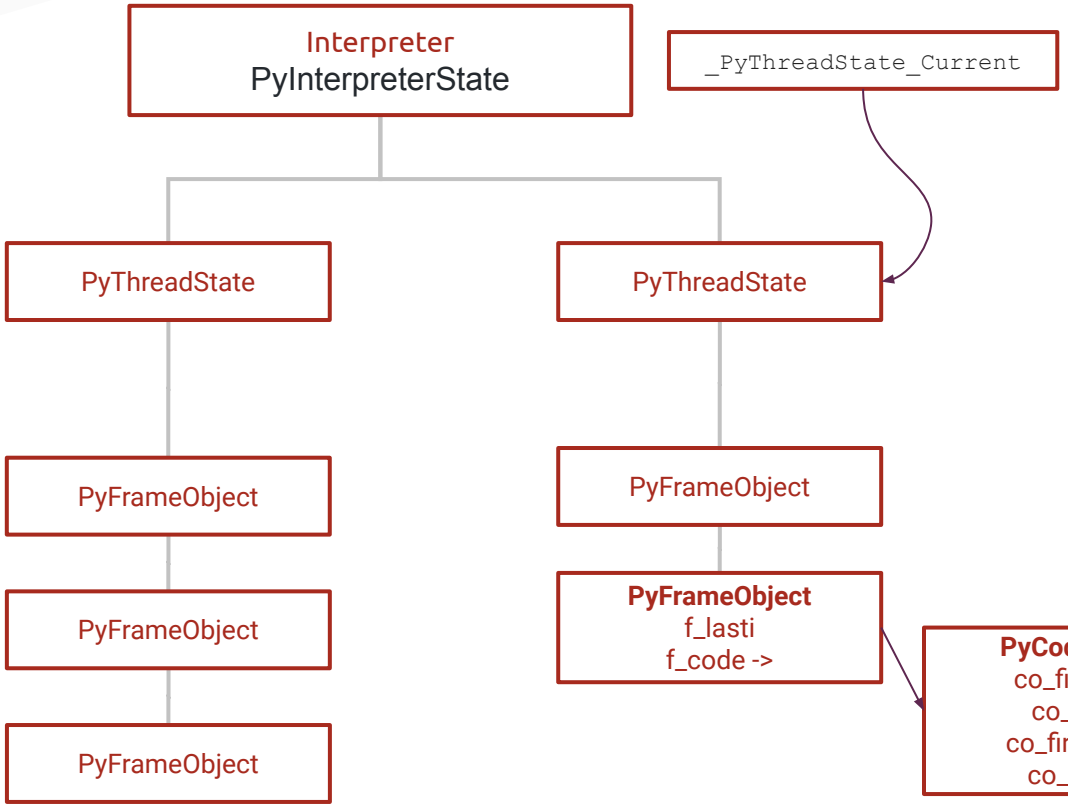
External profiling

How do we get a stack trace?



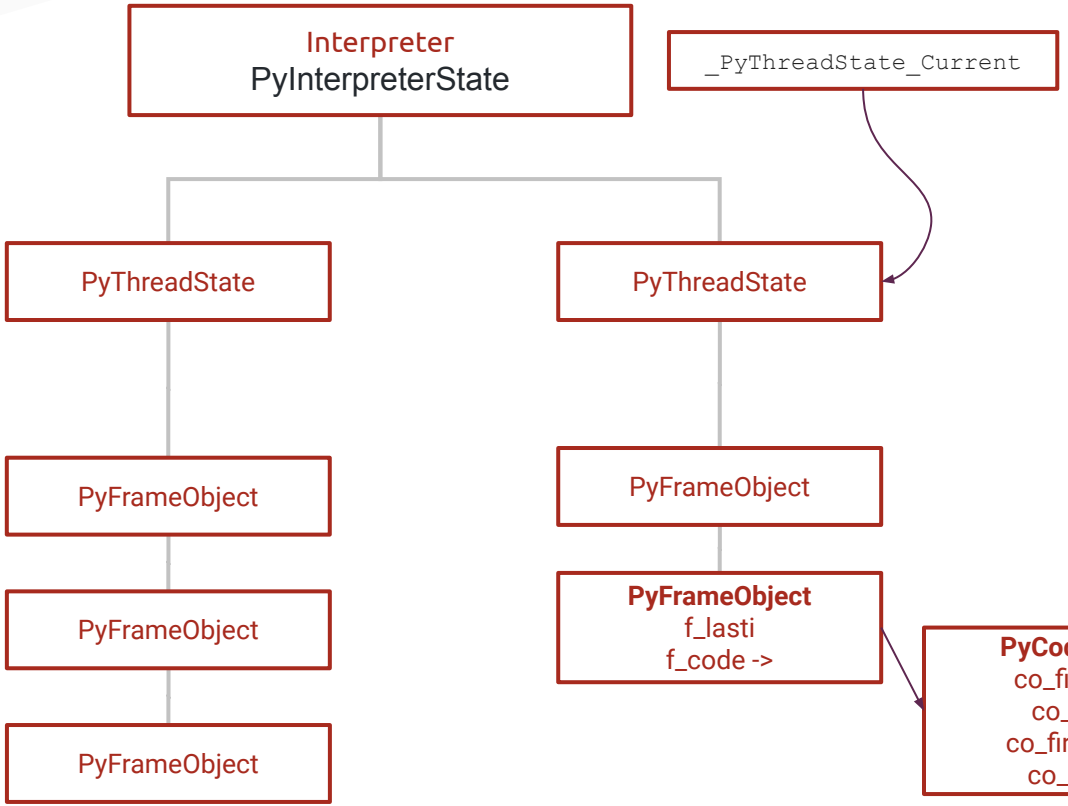

```
struct PyInterpreterState {
        PyInterpreterState *next
        PyThreadState *tstate_head

        int64_t id
        int64_t id_refcount
        PyThread_type_lock id_mutex

        ....
}
```

How do we get a stack trace?



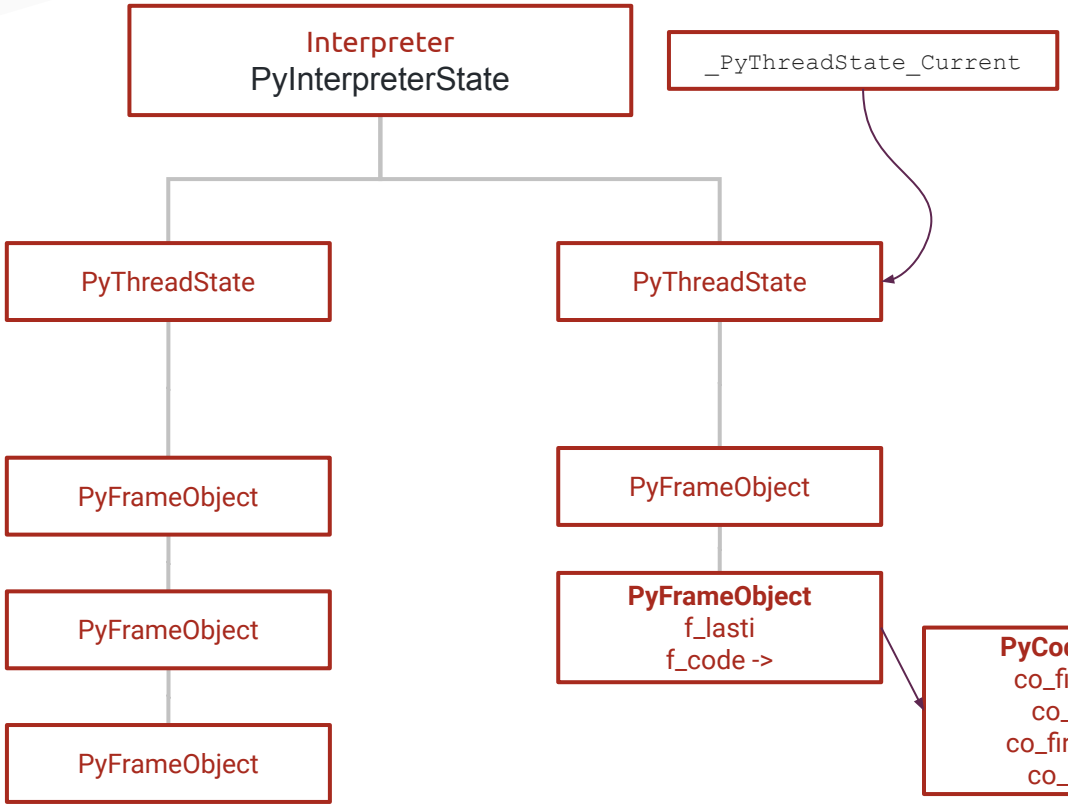

```
struct PyThreadState {
        PyThreadState *prev
        PyThreadState *next
        PyInterpreterState *interp

        PyFrameObject *frame
        int recursion_depth
        char overflowed;
        ....
        Py_tracefunc c_profilefunc;
        ...
}
```

How do we get a stack trace?



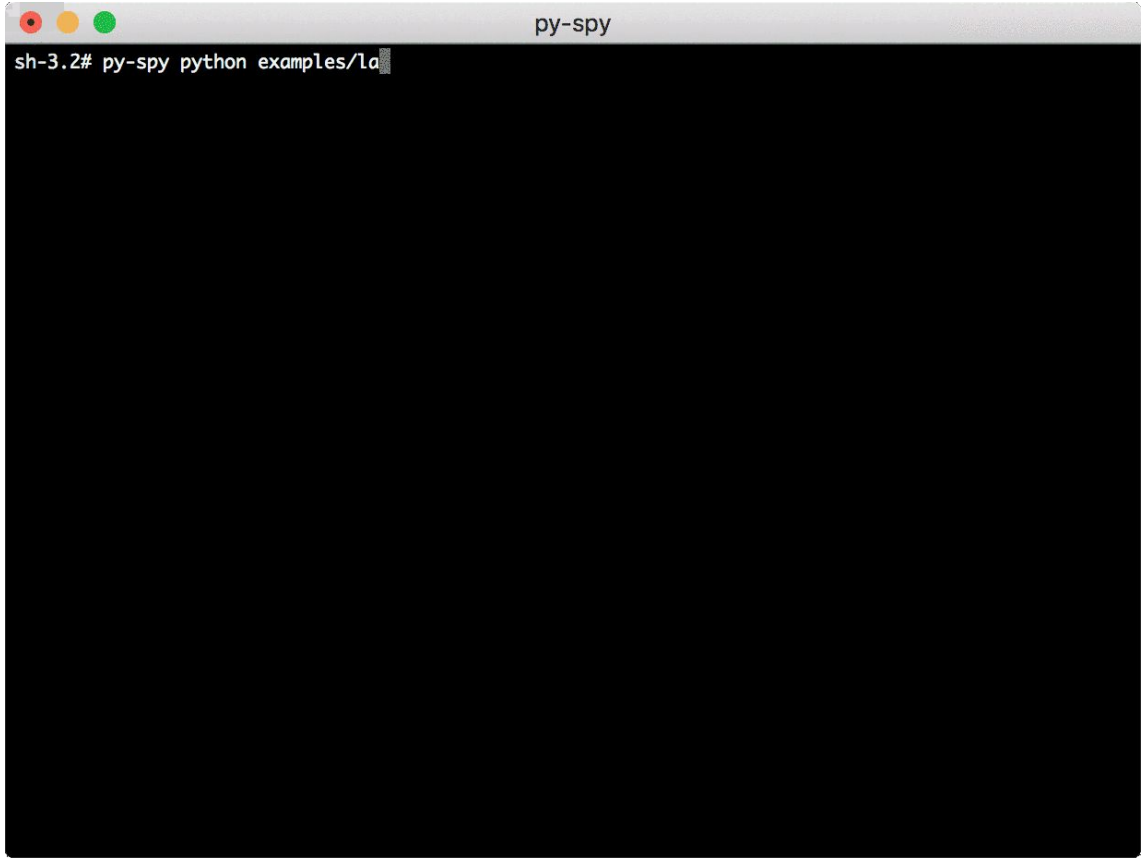

```
PyFrameObject {
        PyFrameObject *back
        PyCodeObject *f_code
        ...
        int f_lasti
}
```

How do we get a stack trace?




```
PyCodeObject {
        int co_argcount
        int co_kwonlyargcount
        int co_nlocals
        int co_stacksize
        int co_flags
        int co_firstlineno
        …
        PyObject *co_filename
        PyObject *co_name
        PyObject *co_lnotab
}
```

py-spy -- python3 examples/lastfm.py



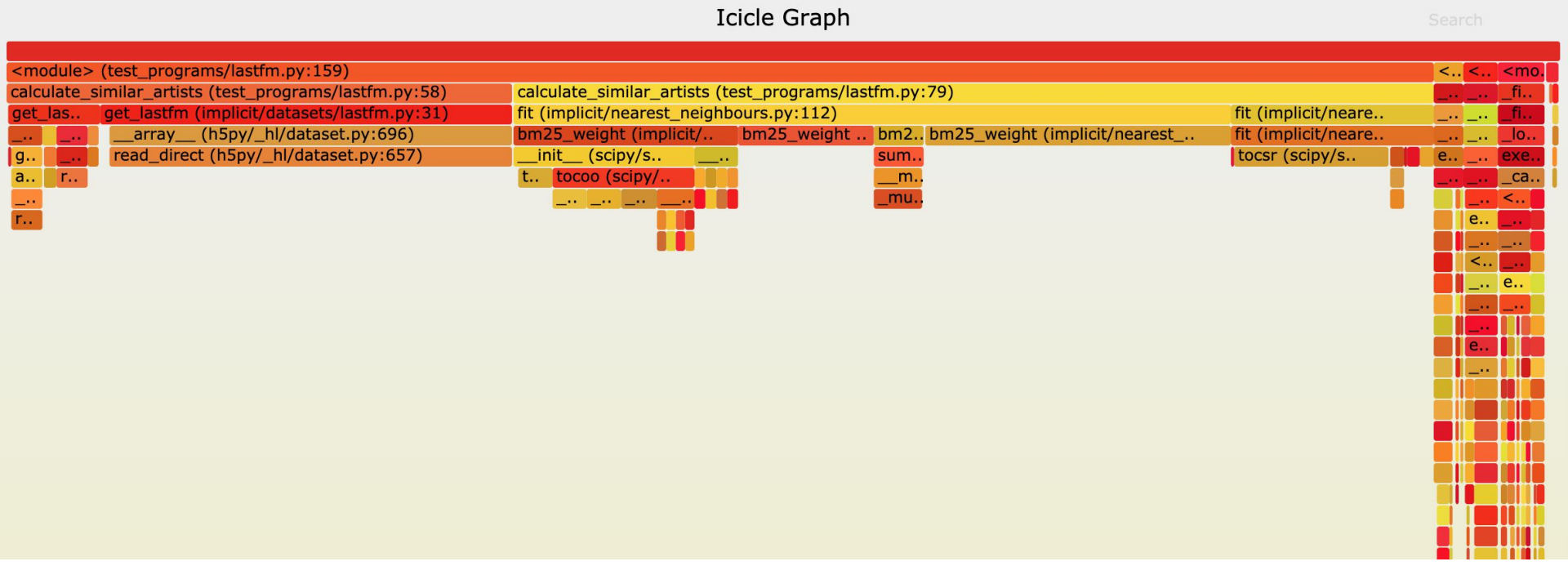
py-spy oneshot

py-spy --dump -p PID

```
/usr/bin/python3.7
Python version 3.7.4
Thread 0x7F5303C4C740 (active)
     func3 (ebpf_example.py:11)
     func2 (ebpf_example.py:17)
     func1 (ebpf_example.py:6)
     <module> (ebpf_example.py:20)
```

py-spy flamegraphs

py-spy --flame flamegraph.svg --duration 60 -- python3 examples/lastfm.py



py-spy history

pyflame

Evan Klitzke (Uber)
September 2016

Rb-spy

Julia Evans
April 2018

Py-spy

Ben Frederickson
August 2018

Tracing

PySnooper

https://github.com/cool-RR/pysnooper

```
import pysnooper

@pysnooper.snoop()
def number_to_bits(number):
    if number:
        bits = []
        while number:
            number, remainder =  divmod(number, 2)
            bits.insert( 0, remainder)
        return bits
    else:
        return [0]

  number_to_bits( 6)
```

PySnooper

```
Starting var:.. number = 6
15:29:11.327032 call         4 def number_to_bits(number):
15:29:11.327032 line         5     if number:
15:29:11.327032 line         6         bits = []
New var:....... bits = []
15:29:11.327032 line         7         while number:
15:29:11.327032 line         8             number, remainder = divmod(number, 2)
New var:....... remainder = 0
Modified var:.. number = 3
15:29:11.327032 line         9             bits.insert(0, remainder)
Modified var:.. bits = [0]
15:29:11.327032 line         7         while number:
15:29:11.327032 line         8             number, remainder = divmod(number, 2)
Modified var:.. number = 1
Modified var:.. remainder = 1
15:29:11.327032 line         9             bits.insert(0, remainder)
Modified var:.. bits = [1, 0]
15:29:11.327032 line         7         while number:
15:29:11.327032 line         8             number, remainder = divmod(number, 2)
Modified var:.. number = 0
15:29:11.327032 line         9             bits.insert(0, remainder)
Modified var:.. bits = [1, 1, 0]
```

eBPF

```
lathiat@optane ~$ sudo tplist-bpfcc -l /usr/bin/python3.8

/usr/bin/python3.8 python:gc__start
/usr/bin/python3.8 python:gc__done
/usr/bin/python3.8 python:audit
/usr/bin/python3.8 python:import__find__load__done
/usr/bin/python3.8 python:import__find__load__start
/usr/bin/python3.8 python:function__return
/usr/bin/python3.8 python:line
/usr/bin/python3.8 python:function__entry
```

bpftrace

```
lathiat@optane ~$ sudo bpftrace -e \
'usdt:/usr/bin/python3.8:function__entry {
  printf("%d: %s:%d %s()\n", pid, str(arg0), arg2, str(arg1)) }' \
 -p $(pidof python3.8)
```

Attaching 1 probe...
26600: ebpf_example.py:5 func1()
26600: ebpf_example.py:13 func2()
26600: ebpf_example.py:9 func3()
26600: ebpf_example.py:13 func2()
26600: ebpf_example.py:5 func1()

bpftrace

```
lathiat@optane ~$ sudo bpftrace -e \
'usdt:/usr/bin/python3.8:function__entry /str(arg1) == "test1"/ {
  printf("%d: %s:%d %s()\n", pid, str(arg0), arg2, str(arg1)) }' \
 -p $(pidof python3.8)
```

Attaching 1 probe...
26600: ebpf_example.py:5 func1()
26600: ebpf_example.py:5 func1()

bpftrace

```
lathiat@optane ~$ sudo bpftrace -e \
'usdt:/usr/bin/python3.8:function__entry {
  @[str(arg1)] = count() }' \
 -p $(pidof python3.8)
```

Attaching 1 probe...
@[func1]: 2
@[func3]: 2
@[func2]: 5

bpfcc-tools

```
lathiat@optane ~$ sudo pythonflow-bpfcc $(pidof python3.8)
Tracing method calls in python process 25972... Ctrl-C to quit.
CPU PID    TID    TIME(us) METHOD
2   25972  25972  3.352    -> ebpf_example.py.func1
2   25972  25972  3.352      -> ebpf_example.py.func2
2   25972  25972  3.352        -> ebpf_example.py.func3
2   25972  25972  4.353        <- ebpf_example.py.func3
2   25972  25972  4.353      <- ebpf_example.py.func2
2   25972  25972  4.353      -> ebpf_example.py.func2
2   25972  25972  4.353      <- ebpf_example.py.func2
2   25972  25972  4.353    <- ebpf_example.py.func1
```

Revise

- Profilers can help with “What is my program doing now?”
- We can attach a profiler with no pre-preparation and minimal overhead
- Tracing (py-snooper) allows us to annoate a specific execution for later analysis
- Tracing (bpftrace) lets us do fast analysis in-kernel

twitter.com/lathiat

lathiat.net/talks

Poll Result

twitter.com/lathiat

lathiat.net/talks

Questions