Python Web Performance 101: Uncovering the root causes

By Alex Ptakhin

Senior Software Engineer at Prestatech GmbH, Berlin

Github

Agenda

- CPU tools
- RAM tools

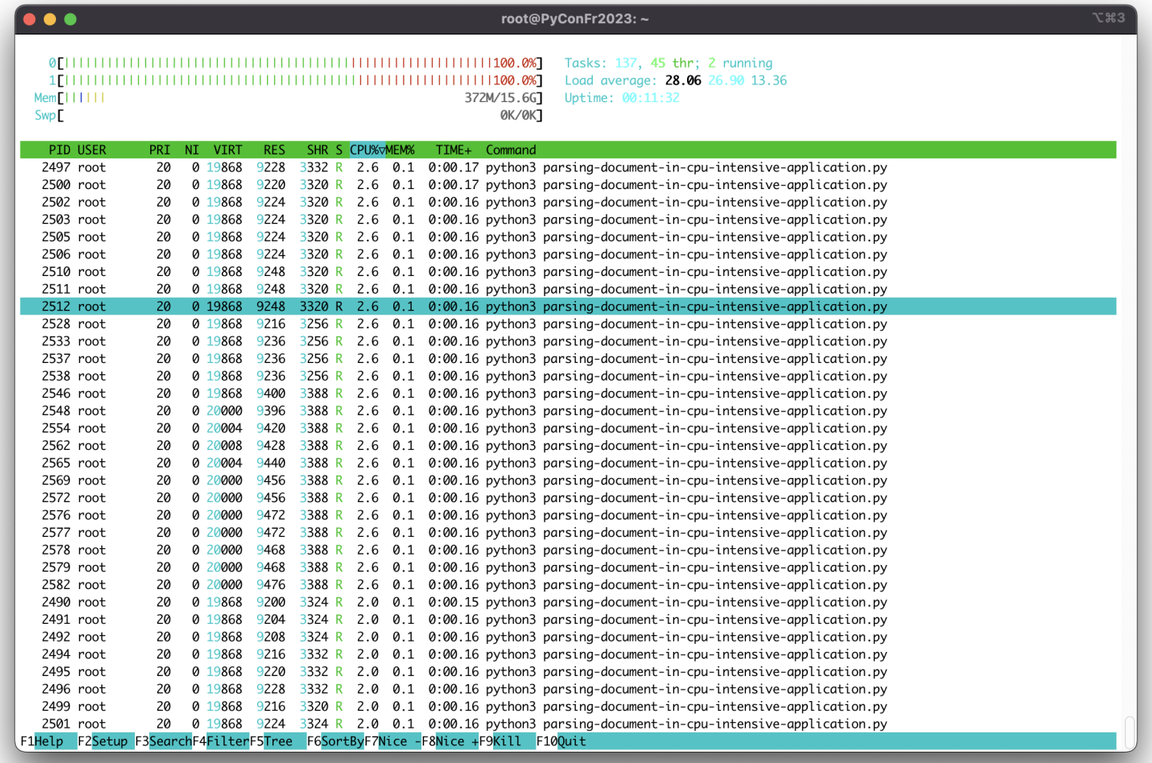
Agenda

- CPU tools
- RAM tools
- Briefly IO
- Tracing

Who at least once used timeit, time.perf_counter(), CPU or memory usage profilers?

Image by Foundry Co from Pixabay

htop


root@PyConFr2023: ~
0[100.0%] Tasks: 137, 45 thr; 2 running
1[100.0%] Load average: 28.06 26.90 13.36
Mem[372M/15.6G] Uptime: 00:11:32
Swp[0K/0K]
PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
2497 root 20 0 19868 9228 3332 R 2.6 0.1 0:00.17 python3 parsing-document-in-cpu-intensive-application.py
2500 root 20 0 19868 9220 3320 R 2.6 0.1 0:00.17 python3 parsing-document-in-cpu-intensive-application.py
2502 root 20 0 19868 9224 3320 R 2.6 0.1 0:00.16 python3 parsing-document-in-cpu-intensive-application.py
2503 root 20 0 19868 9224 3320 R 2.6 0.1 0:00.16 python3 parsing-document-in-cpu-intensive-application.py
2505 root 20 0 19868 9224 3320 R 2.6 0.1 0:00.16 python3 parsing-document-in-cpu-intensive-application.py
2506 root 20 0 19868 9224 3320 R 2.6 0.1 0:00.16 python3 parsing-document-in-cpu-intensive-application.py
2510 root 20 0 19868 9248 3320 R 2.6 0.1 0:00.16 python3 parsing-document-in-cpu-intensive-application.py
2511 root 20 0 19868 9248 3320 R 2.6 0.1 0:00.16 python3 parsing-document-in-cpu-intensive-application.py
2512 root 20 0 19868 9248 3320 R 2.6 0.1 0:00.16 python3 parsing-document-in-cpu-intensive-application.py
2528 root 20 0 19868 9216 3256 R 2.6 0.1 0:00.16 python3 parsing-document-in-cpu-intensive-application.py
2533 root 20 0 19868 9236 3256 R 2.6 0.1 0:00.16 python3 parsing-document-in-cpu-intensive-application.py
2537 root 20 0 19868 9236 3256 R 2.6 0.1 0:00.16 python3 parsing-document-in-cpu-intensive-application.py
2538 root 20 0 19868 9236 3256 R 2.6 0.1 0:00.16 python3 parsing-document-in-cpu-intensive-application.py
2546 root 20 0 19868 9400 3388 R 2.6 0.1 0:00.16 python3 parsing-document-in-cpu-intensive-application.py
2548 root 20 0 20000 9396 3388 R 2.6 0.1 0:00.16 python3 parsing-document-in-cpu-intensive-application.py
2554 root 20 0 20004 9420 3388 R 2.6 0.1 0:00.16 python3 parsing-document-in-cpu-intensive-application.py
2562 root 20 0 20008 9428 3388 R 2.6 0.1 0:00.16 python3 parsing-document-in-cpu-intensive-application.py
2565 root 20 0 20004 9440 3388 R 2.6 0.1 0:00.16 python3 parsing-document-in-cpu-intensive-application.py
2569 root 20 0 20000 9456 3388 R 2.6 0.1 0:00.16 python3 parsing-document-in-cpu-intensive-application.py
2572 root 20 0 20000 9456 3388 R 2.6 0.1 0:00.16 python3 parsing-document-in-cpu-intensive-application.py
2576 root 20 0 20000 9472 3388 R 2.6 0.1 0:00.16 python3 parsing-document-in-cpu-intensive-application.py
2577 root 20 0 20000 9472 3388 R 2.6 0.1 0:00.16 python3 parsing-document-in-cpu-intensive-application.py
2578 root 20 0 20000 9468 3388 R 2.6 0.1 0:00.16 python3 parsing-document-in-cpu-intensive-application.py
2579 root 20 0 20000 9468 3388 R 2.6 0.1 0:00.16 python3 parsing-document-in-cpu-intensive-application.py
2582 root 20 0 20000 9476 3388 R 2.6 0.1 0:00.16 python3 parsing-document-in-cpu-intensive-application.py
2490 root 20 0 19868 9200 3324 R 2.0 0.1 0:00.15 python3 parsing-document-in-cpu-intensive-application.py
2491 root 20 0 19868 9204 3324 R 2.0 0.1 0:00.16 python3 parsing-document-in-cpu-intensive-application.py
2492 root 20 0 19868 9208 3324 R 2.0 0.1 0:00.16 python3 parsing-document-in-cpu-intensive-application.py
2494 root 20 0 19868 9216 3332 R 2.0 0.1 0:00.16 python3 parsing-document-in-cpu-intensive-application.py
2495 root 20 0 19868 9220 3332 R 2.0 0.1 0:00.16 python3 parsing-document-in-cpu-intensive-application.py
2496 root 20 0 19868 9228 3332 R 2.0 0.1 0:00.16 python3 parsing-document-in-cpu-intensive-application.py
2499 root 20 0 19868 9216 3320 R 2.0 0.1 0:00.16 python3 parsing-document-in-cpu-intensive-application.py
2501 root 20 0 19868 9224 3324 R 2.0 0.1 0:00.16 python3 parsing-document-in-cpu-intensive-application.py
F1Help F2Setup F3SearchF4FilterF5Tree F6SortByF7Nice -F8Nice +F9Kill F10Quit

Temporary solution

Temporary solution

Scale-up: more CPU, more RAM

CPU

time.perf_counter

```
1  import time
2  from calls import cpu_intensive_call
3
4  start = time.perf_counter()
5
6  cpu_intensive_call(num_iterations=5000000)
7
8  end = time.perf_counter()
9  print('Elapsed seconds: {:.1f}'.format(end - start))
```

time.perf_counter

- Out of box

time.perf_counter

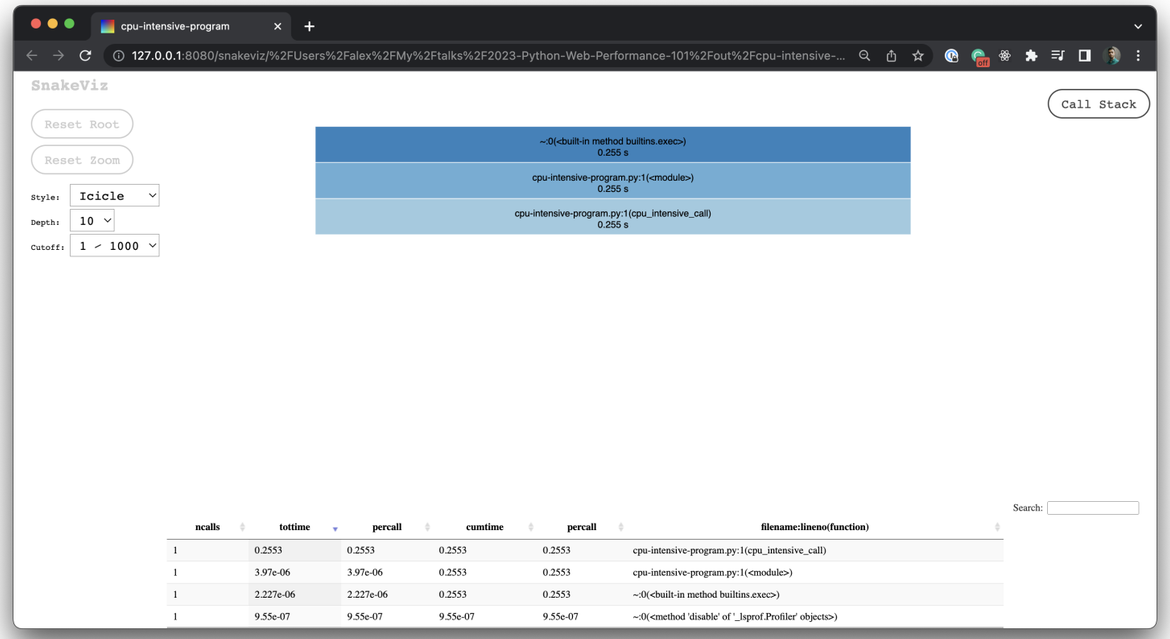
- Out of box
- Need to edit code, no internal details

cProfile

```
1  import cProfile
2  import re
3
4  from calls import cpu_intensive_call
5
6  cProfile.run('cpu_intensive_call(num_iterations=5000000)')
```

cProfile

```
$ python -m cProfile \
    -o out/cpu-intensive-program.prof \
    load/cpu-intensive-program.py
$ snakeviz out/cpu-intensive-program.prof
```



cProfile

- Out of box
- Internal details timings

cProfile

- Out of box
- Internal details timings
- Have visualize extensions

CPU Profilers

- Perfect! Is this all?

CPU Profilers

- Perfect! Is this all?
- Not exactly. Measuring something can change a behaviour of the system

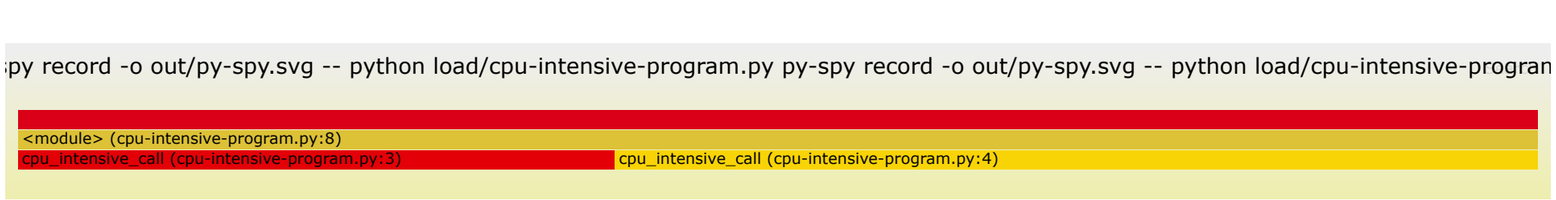
CPU Profilers

- Perfect! Is this all?
- Not exactly. Measuring something can change a behaviour of the system
- Let's take a look to sampling profilers

py-spy

```python
def cpu_intensive_call(*, num_iterations):
    x = 1
    for _ in range(num_iterations):
        x*x

if __name__ == '__main__':
    cpu_intensive_call(num_iterations=5000000)
```

```
$ py-spy record -o out/py-spy.svg -- python load/cpu-intensive-program.py
```



py-spy

- Sampling profiler

py-spy

- Sampling profiler
- Requires development environment

Bonus: yappi for asyncio

```
import asyncio
import yappi
from calls import cpu_intensive_call

async def foo():
    await asyncio.sleep(1.0)
    await baz()
    cpu_intensive_call(num_iterations=10000000)
    await asyncio.sleep(0.5)

async def baz():
    await asyncio.sleep(1.0)

yappi.set_clock_type('wall')
with yappi.run():
    asyncio.run(foo())
yappi.get_func_stats().print_all()
yappi.get_func_stats()._save_as_PSTAT('out/asyncio_yappi.prof')
```

```
$ python load/asyncio_yappi.py
```

```
Clock type: WALL
Ordered by: totaltime, desc

name                                  ncall  tsub      ttot      tavg
```

```
..ython3.10/asyncio/runners.py:8 run  1      0.000052  3.115420  3.115420
..lectorEventLoop.run_until_complete  3      0.000124  3.114527  1.038176
..._UnixSelectorEventLoop.run_forever  3      0.000067  3.114223  1.038074
..4 _UnixSelectorEventLoop._run_once  12     0.000375  3.114111  0.259509
..ce-101/load/asyncio_yappi.py:5 foo  1      0.000024  3.111857  3.111857
..ctors.py:554 KqueueSelector.select  12     0.000223  2.502546  0.208545
..hon3.10/asyncio/tasks.py:593 sleep  3      2.502370  2.502526  0.834175
..e-101/load/asyncio_yappi.py:11 baz  1      0.000005  1.000425  1.000425
..0/asyncio/events.py:78 Handle._run  12     0.000053  0.611028  0.050919
..-101/calls.py:1 cpu_intensive_call  1      0.609302  0.609302  0.609302
..lectorEventLoop.shutdown_asyncgens  1      0.001139  0.001147  0.001147
..yncio/events.py:780 new_event_loop  1      0.000045  0.000425  0.000425
..aultEventLoopPolicy.new_event_loop  1      0.000008  0.000338  0.000338
..63 _UnixSelectorEventLoop.__init__  1      0.000004  0.000330  0.000330
..49 _UnixSelectorEventLoop.__init__  1      0.000024  0.000326  0.000326
..xSelectorEventLoop._make_self_pipe  1      0.000024  0.000173  0.000173
..io/runners.py:55 _cancel_all_tasks  1      0.000006  0.000173  0.000173
..py:67 _UnixSelectorEventLoop.close  1      0.000013  0.000170  0.000170
..3.10/asyncio/tasks.py:42 all_tasks  1      0.000051  0.000167  0.000167
..py:82 _UnixSelectorEventLoop.close  1      0.000016  0.000157  0.000157
..asyncio/tasks.py:610 ensure_future  3      0.000008  0.000134  0.000045
..syncio/tasks.py:618 _ensure_future  3      0.000021  0.000126  0.000042
..3 _UnixSelectorEventLoop.call_soon  9      0.000030  0.000124  0.000014
..py:696 _UnixSelectorEventLoop.time  18     0.000099  0.000116  0.000006
..SelectorEventLoop._close_self_pipe  1      0.000023  0.000112  0.000112
.._weakrefset.py:63 WeakSet.__iter__  1      0.000033  0.000110  0.000110
.. _UnixSelectorEventLoop.call_later  3      0.000026  0.000096  0.000032
.._UnixSelectorEventLoop.create_task  3      0.000049  0.000096  0.000032
.._UnixSelectorEventLoop._add_reader  1      0.000018  0.000089  0.000089
.. _UnixSelectorEventLoop._call_soon  9      0.000051  0.000087  0.000010
..y:309 _set_result_unless_cancelled  3      0.000015  0.000086  0.000029
..ents.py:184 _run_until_complete_cb  3      0.000042  0.000073  0.000024
..yncio/events.py:31 Handle.__init__  13     0.000055  0.000070  0.000005
..91 _UnixSelectorEventLoop.__init__  1      0.000024  0.000066  0.000066
```

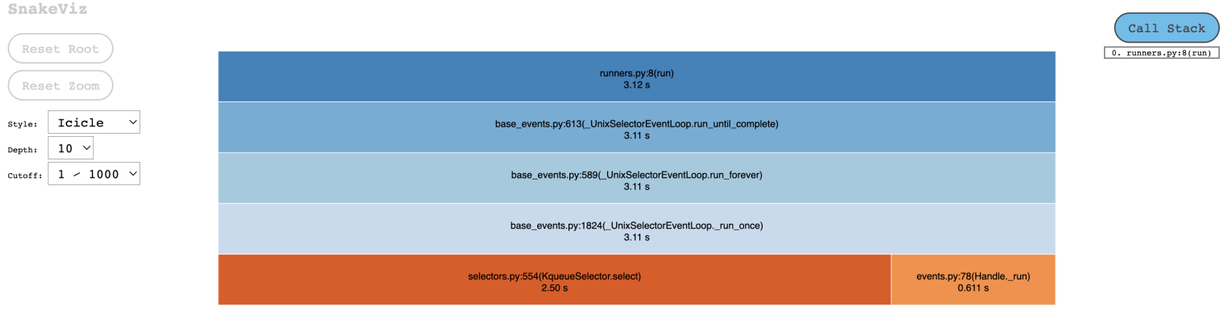
..727 _UnixSelectorEventLoop.call_at	3	0.000025	0.000063	0.000021
..ixSelectorEventLoop._remove_reader	1	0.000014	0.000055	0.000055
..ython3.10/socket.py:594 socketpair	1	0.000018	0.000053	0.000053
..et.py:21 _IterationGuard.__enter__	1	0.000046	0.000047	0.000047
..vents.py:736 get_event_loop_policy	3	0.000002	0.000043	0.000014
..nts.py:728 _init_event_loop_policy	1	0.000019	0.000041	0.000041
..o/events.py:148 TimerHandle.cancel	3	0.000027	0.000039	0.000013
..hon3.10/socket.py:498 socket.close	2	0.000009	0.000034	0.000017
..ors.py:517 KqueueSelector.register	1	0.000010	0.000034	0.000034
..events.py:103 TimerHandle.__init__	3	0.000015	0.000033	0.000011
..yncio/events.py:775 set_event_loop	2	0.000007	0.000031	0.000016
..tors.py:181 KqueueSelector.get_key	2	0.000010	0.000030	0.000015
..s.py:533 KqueueSelector.unregister	1	0.000010	0.000028	0.000028
..ging/__init__.py:1455 Logger.debug	1	0.000004	0.000027	0.000027
..set.py:27 _IterationGuard.__exit__	1	0.000016	0.000027	0.000027
..io/coroutines.py:18 _is_debug_mode	1	0.000009	0.000026	0.000026
..0/socket.py:494 socket._real_close	2	0.000008	0.000025	0.000012
..syncio/base_futures.py:14 isfuture	6	0.000012	0.000025	0.000004
..aultEventLoopPolicy.set_event_loop	2	0.000015	0.000023	0.000012
..init__.py:1724 Logger.isEnabledFor	1	0.000009	0.000023	0.000023
..ixSelectorEventLoop._check_running	6	0.000016	0.000023	0.000004
..thon3.10/_weakrefset.py:39 _remove	3	0.000013	0.000021	0.000007
..ors.py:510 KqueueSelector.__init__	1	0.000014	0.000020	0.000020
..py:70 _SelectorMapping.__getitem__	2	0.000009	0.000019	0.000009
..ectors.py:579 KqueueSelector.close	1	0.000008	0.000019	0.000019
..ncio/coroutines.py:177 iscoroutine	4	0.000017	0.000018	0.000005
..ollections_abc.py:816 _Environ.get	1	0.000007	0.000017	0.000017
..nixSelectorEventLoop.create_future	3	0.000015	0.000017	0.000006
..216 KqueueSelector._fileobj_lookup	4	0.000010	0.000016	0.000004
..y:105 WeakValueDictionary.__init__	1	0.000008	0.000016	0.000016
..nixSelectorEventLoop._check_closed	22	0.000014	0.000014	0.000001
..xSelectorEventLoop._process_events	12	0.000014	0.000014	0.000001
..io/events.py:64 TimerHandle.cancel	4	0.000012	0.000014	0.000003
..7 _UnixSelectorEventLoop.get_debug	23	0.000014	0.000014	0.000001

```
..ors.py:235 KqueueSelector.register  1      0.000007  0.000014  0.000014
..s.py:248 KqueueSelector.unregister  1      0.000006  0.000012  0.000012
..3.10/socket.py:220 socket.__init__  2      0.000011  0.000011  0.000005
..nixDefaultEventLoopPolicy.__init__  1      0.000005  0.000011  0.000011
..3.10/_weakrefset.py:86 WeakSet.add  3      0.000010  0.000011  0.000004
...10/os.py:675 _Environ.__getitem__  1      0.000005  0.000010  0.000010
..set.py:53 WeakSet._commit_removals  1      0.000009  0.000010  0.000010
..oop._set_coroutine_origin_tracking  6      0.000009  0.000009  0.000002
..y:659 _UnixSelectorEventLoop.close  1      0.000007  0.000009  0.000009
..aultEventLoopPolicy.set_event_loop  2      0.000008  0.000008  0.000004
...py:290 WeakValueDictionary.update  1      0.000006  0.000008  0.000008
..0/asyncio/futures.py:297 _get_loop  3      0.000008  0.000008  0.000003
../_weakrefset.py:72 WeakSet.__len__  2      0.000006  0.000008  0.000004
..ectors.py:269 KqueueSelector.close  1      0.000007  0.000008  0.000008
..ors.py:210 KqueueSelector.__init__  1      0.000005  0.000006  0.000006
..nixDefaultEventLoopPolicy.__init__  1      0.000006  0.000006  0.000006
...10/selectors.py:21 _fileobj_to_fd  4      0.000004  0.000006  0.000002
..._bootstrap>:1053 _handle_fromlist  1      0.000004  0.000006  0.000006
..gging/__init__.py:228 _releaseLock  1      0.000005  0.000006  0.000006
..10/lib/python3.10/os.py:755 encode  1      0.000004  0.000005  0.000005
:1                    1      0.000004  0.000005  0.000005
..gging/__init__.py:219 _acquireLock  1      0.000005  0.000005  0.000005
..._bootstrap>:404 ModuleSpec.parent  1      0.000004  0.000005  0.000005
..._weakrefset.py:37 WeakSet.__init__  1      0.000004  0.000004  0.000004
..py:651 _UnixSelectorEventLoop.stop  3      0.000004  0.000004  0.000001
.. _UnixSelectorEventLoop.is_running  9      0.000004  0.000004  0.000000
..0 _UnixSelectorEventLoop.set_debug  1      0.000003  0.000004  0.000004
..set.py:17 _IterationGuard.__init__  1      0.000003  0.000003  0.000003
..ventLoop.shutdown_default_executor  1      0.000002  0.000002  0.000002
.._.py:1710 Logger.getEffectiveLevel  1      0.000002  0.000002  0.000002
..2 _UnixSelectorEventLoop.is_closed  3      0.000002  0.000002  0.000001
..3.10/asyncio/tasks.py:61   1      0.000002  0.000002  0.000002
..686 _UnixSelectorEventLoop.__del__  1      0.000000  0.000001  0.000001
..rs.py:64 _SelectorMapping.__init__  1      0.000001  0.000001  0.000001
```

```
..tors.py:273 KqueueSelector.get_map  2      0.000001  0.000001  0.000000
..rEventLoop._timer_handle_cancelled  3      0.000001  0.000001  0.000000
..g/__init__.py:1307 Manager.disable  1      0.000001  0.000001  0.000001
.. _GeneratorContextManager.__exit__  1      0.000000  0.000000  0.000000
```

Bonus: yappi for asyncio

```
$ python load/asyncio_yappi.py > out/asyncio_yappi.txt
$ snakeviz out/asyncio_yappi.prof
```



Search:

ncalls	tottime	percall	cumtime	percall	filename:lineno(function)
3	2.502	0.8341	2.503	0.8342	tasks.py:593(sleep)
1	0.6093	0.6093	0.6093	0.6093	calls.py:1(cpu_intensive_call)
1	0.001139	0.001139	0.001147	0.001147	base_events.py:535(_UnixSelectorEventLoop.shutdown_asyncgens)
12	0.000375	3.125e-05	3.114	0.2595	base_events.py:1824(_UnixSelectorEventLoop._run_once)
12	0.000223	1.858e-05	2.503	0.2085	selectors.py:554(KqueueSelector.select)
3	0.000124	4.133e-05	3.115	1.038	base_events.py:613(_UnixSelectorEventLoop.run_until_complete)
18	9.9e-05	5.5e-06	0.000116	6.444e-06	base_events.py:696(_UnixSelectorEventLoop.time)

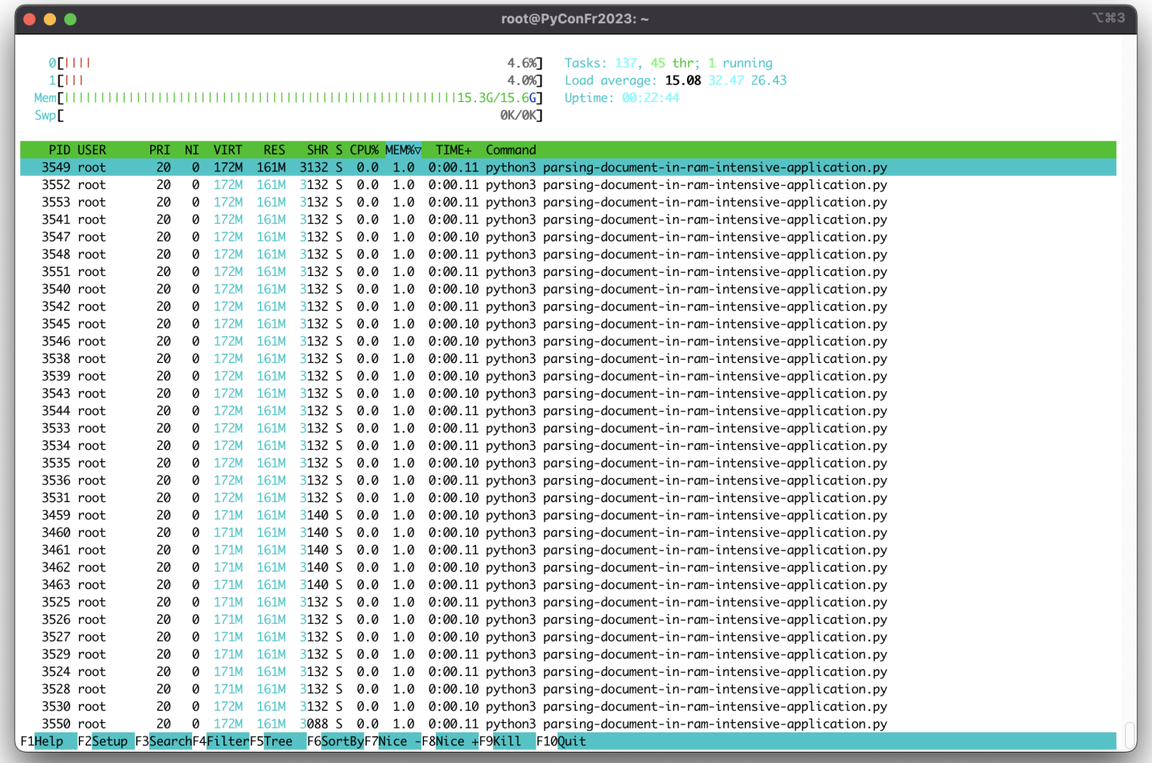
Bonus: yappi for asyncio

- Supports asynchronous execution
- Different clock modes

Problem found

After a few days

htop


root@PyConFr2023: ~
0[|||| 4.6%] Tasks: 137, 45 thr; 1 running
1[||| 4.0%] Load average: 15.08 32.47 26.43
Mem[|||||||||||||||||||||||||||||||||||||||||||||||||||15.3G/15.6G] Uptime: 00:22:44
Swp[0K/0K]
PID USER PRI NI VIRT RES SHR S CPU% MEM%▽ TIME+ Command
3549 root 20 0 172M 161M 3132 S 0.0 1.0 0:00.11 python3 parsing-document-in-ram-intensive-application.py
3552 root 20 0 172M 161M 3132 S 0.0 1.0 0:00.11 python3 parsing-document-in-ram-intensive-application.py
3553 root 20 0 172M 161M 3132 S 0.0 1.0 0:00.11 python3 parsing-document-in-ram-intensive-application.py
3541 root 20 0 172M 161M 3132 S 0.0 1.0 0:00.11 python3 parsing-document-in-ram-intensive-application.py
3547 root 20 0 172M 161M 3132 S 0.0 1.0 0:00.10 python3 parsing-document-in-ram-intensive-application.py
3548 root 20 0 172M 161M 3132 S 0.0 1.0 0:00.11 python3 parsing-document-in-ram-intensive-application.py
3551 root 20 0 172M 161M 3132 S 0.0 1.0 0:00.11 python3 parsing-document-in-ram-intensive-application.py
3540 root 20 0 172M 161M 3132 S 0.0 1.0 0:00.10 python3 parsing-document-in-ram-intensive-application.py
3542 root 20 0 172M 161M 3132 S 0.0 1.0 0:00.11 python3 parsing-document-in-ram-intensive-application.py
3545 root 20 0 172M 161M 3132 S 0.0 1.0 0:00.10 python3 parsing-document-in-ram-intensive-application.py
3546 root 20 0 172M 161M 3132 S 0.0 1.0 0:00.10 python3 parsing-document-in-ram-intensive-application.py
3538 root 20 0 172M 161M 3132 S 0.0 1.0 0:00.11 python3 parsing-document-in-ram-intensive-application.py
3539 root 20 0 172M 161M 3132 S 0.0 1.0 0:00.10 python3 parsing-document-in-ram-intensive-application.py
3543 root 20 0 172M 161M 3132 S 0.0 1.0 0:00.10 python3 parsing-document-in-ram-intensive-application.py
3544 root 20 0 172M 161M 3132 S 0.0 1.0 0:00.11 python3 parsing-document-in-ram-intensive-application.py
3533 root 20 0 172M 161M 3132 S 0.0 1.0 0:00.11 python3 parsing-document-in-ram-intensive-application.py
3534 root 20 0 172M 161M 3132 S 0.0 1.0 0:00.11 python3 parsing-document-in-ram-intensive-application.py
3535 root 20 0 172M 161M 3132 S 0.0 1.0 0:00.10 python3 parsing-document-in-ram-intensive-application.py
3536 root 20 0 172M 161M 3132 S 0.0 1.0 0:00.11 python3 parsing-document-in-ram-intensive-application.py
3531 root 20 0 172M 161M 3132 S 0.0 1.0 0:00.10 python3 parsing-document-in-ram-intensive-application.py
3459 root 20 0 171M 161M 3140 S 0.0 1.0 0:00.10 python3 parsing-document-in-ram-intensive-application.py
3460 root 20 0 171M 161M 3140 S 0.0 1.0 0:00.10 python3 parsing-document-in-ram-intensive-application.py
3461 root 20 0 171M 161M 3140 S 0.0 1.0 0:00.11 python3 parsing-document-in-ram-intensive-application.py
3462 root 20 0 171M 161M 3140 S 0.0 1.0 0:00.10 python3 parsing-document-in-ram-intensive-application.py
3463 root 20 0 171M 161M 3140 S 0.0 1.0 0:00.11 python3 parsing-document-in-ram-intensive-application.py
3525 root 20 0 171M 161M 3132 S 0.0 1.0 0:00.11 python3 parsing-document-in-ram-intensive-application.py
3526 root 20 0 171M 161M 3132 S 0.0 1.0 0:00.10 python3 parsing-document-in-ram-intensive-application.py
3527 root 20 0 171M 161M 3132 S 0.0 1.0 0:00.10 python3 parsing-document-in-ram-intensive-application.py
3529 root 20 0 171M 161M 3132 S 0.0 1.0 0:00.11 python3 parsing-document-in-ram-intensive-application.py
3524 root 20 0 171M 161M 3132 S 0.0 1.0 0:00.11 python3 parsing-document-in-ram-intensive-application.py
3528 root 20 0 171M 161M 3132 S 0.0 1.0 0:00.10 python3 parsing-document-in-ram-intensive-application.py
3530 root 20 0 172M 161M 3132 S 0.0 1.0 0:00.10 python3 parsing-document-in-ram-intensive-application.py
3550 root 20 0 172M 161M 3088 S 0.0 1.0 0:00.11 python3 parsing-document-in-ram-intensive-application.py
F1Help F2Setup F3SearchF4FilterF5Tree F6SortByF7Nice -F8Nice +F9Kill F10Quit

RAM

Temporary solution

Temporary solution

Restart every N requests

sys.getsizeof

```
1  import sys
2
3  print(f'Empty dict size: {sys.getsizeof({})}')
4  print(f'Empty list size: {sys.getsizeof([])}')
5  print(f'Empty set size: {sys.getsizeof(set())}')
```

tracemalloc

```python
import tracemalloc

def ram_intensive_dummy_call() -> None:
    a = [1] * (10 ** 6)
    b = [2] * (2 * 10 ** 7)
    del b
    return a

tracemalloc.start()

snapshot1 = tracemalloc.take_snapshot()
ram_intensive_dummy_call()
snapshot2 = tracemalloc.take_snapshot()

top_stats = snapshot2.compare_to(snapshot1, 'lineno')
print("[ Top 10 differences ]")
for stat in top_stats[:10]:
    print(stat)
```

memory-profiler

```python
@profile
def dummy_ram_intensive_call():
    a = [1] * (10 ** 6)
    b = [2] * (2 * 10 ** 7)
    del b
    return a

if __name__ == '__main__':
    dummy_ram_intensive_call()
```

```
$ poetry add memory_profiler
$ python -m memory_profiler load/memory_profiler.py


Filename: load/memory_profiler.py

Line #    Mem usage    Increment  Occurrences   Line Contents
=============================================================
     1   15.219 MiB   15.219 MiB           1   @profile
     2                                         def dummy_ram_intensive_call():
     3   22.852 MiB    7.633 MiB           1       a = [1] * (10 ** 6)
     4  175.441 MiB  152.590 MiB           1       b = [2] * (2 * 10 ** 7)
     5   22.852 MiB -152.590 MiB           1       del b
     6   22.852 MiB    0.000 MiB           1       return a
```

memory-profiler

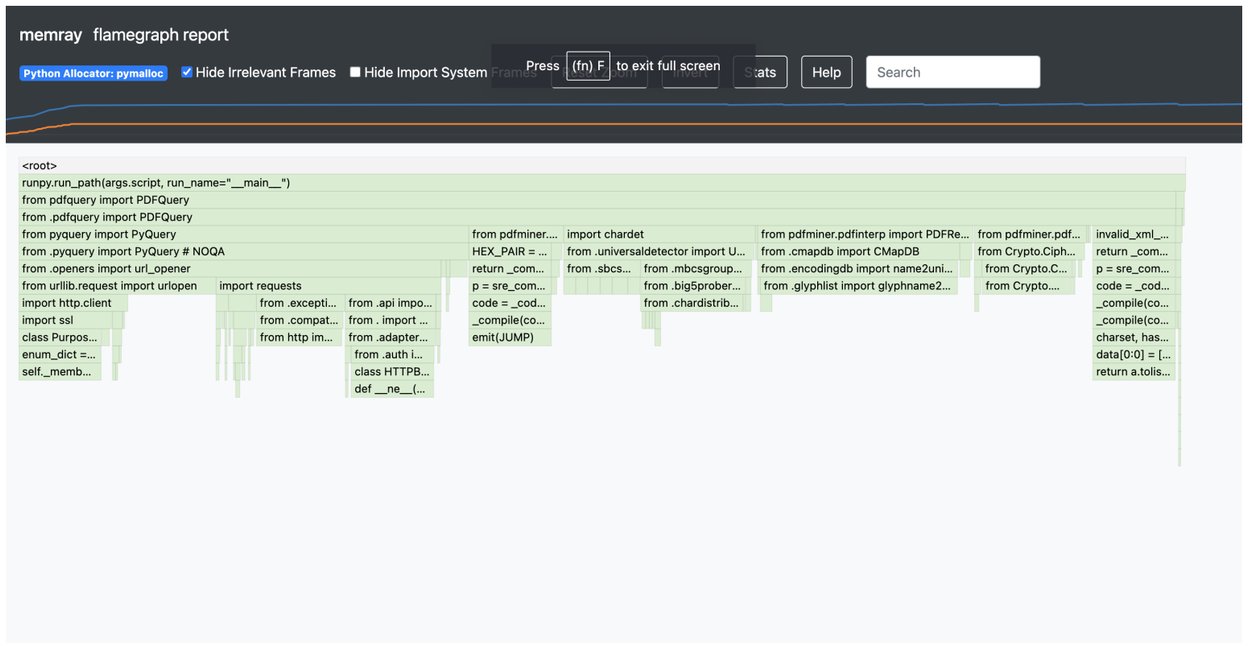
- Requires code changes for the detailed overview

memory-profiler

- Code changes for the detailed overview
- Uses deprecated matplotlib.pylab
- No longer maintained

memray

```
$ poetry add memray
$ memray run -o out/memray.bin load/ram-intensive-program.py
$ memray flamegraph out/memray.bin
$ # ... out/memray-flamegraph-memray.html
```



memray

- Looks promising

OI

General advices

General advices

- Scale up
- Scale out
- Network

Tracing

Open Telemetry

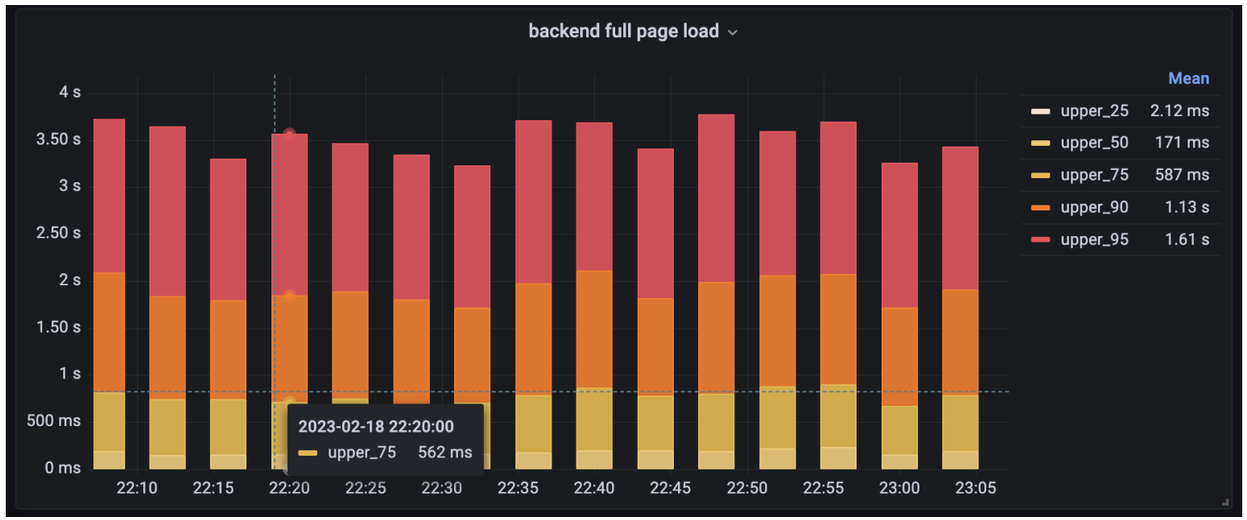
```python
from calls import cpu_intensive_call
from opentelemetry import trace

tracer = trace.get_tracer(__name__)

if __name__ == '__main__':
    with tracer.start_as_current_span("cpu_intensive_call") as child:
        cpu_intensive_call(num_iterations=5000000)
```

Open Telemetry

```python
from otel_helpers import catchtime, init_otel
from opentelemetry import trace, metrics

from calls import cpu_intensive_call

init_otel()

tracer = trace.get_tracer(__name__)
meter = metrics.get_meter(__name__)
execution_time_hgram = meter.create_histogram('execution_time')

with tracer.start_as_current_span("cpu_intensive_application") as parent:
    for x in range(3):
        with tracer.start_as_current_span("cpu_intensive_call") as child, catchtime() as t:
            cpu_intensive_call(num_iterations=5000000)
        execution_time_hgram.record(t())
```

Open Telemetry

Multiply vendors, e.g. Grafana.



3 things to remember

3 things to remember

- Worth to have a chance win some time with resources

3 things to remember

- Worth to have a chance win some time with resources
- Monitor application errors

~~3~~ 4 things to remember

- Worth to have a chance win some time with resources
- Monitor application errors
- Measuring something can change a behaviour of the system
- Tuning is good and remember pure Python is not about the performance

Thank you! Questions?

Gratitudes

- PyScript for empowering the presentation `<py>`
- highlight.js for syntax highlighting
- playwright for help with PDF slides
- Prestatech GmbH for support this talk (and we are hiring)

By Alex Ptakhin, Senior Software Engineer at Prestatech GmbH, Berlin. me@aptakhin.name / github.com/aptakhin / twitter.com/aptakhin / hachyderm.io/@AlexPtakhin / linkedin.com/in/aptakhin

Presentation sources: github.com/aptakhin/talks/2023-Python-Web-Performance-101

Secret reference slide

- https://docs.python.org/3/library/profile.html
- Yappi clock types:
  - CPU Time
  - WALL Time
- https://docs.python.org/3/library/tracemalloc.html
- https://bloomberg.github.io/memray/
- https://opentelemetry.io/ecosystem/vendors/
- init_otel implementantion python source